НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ДИПЛОМНА РОБОТА
на здобуття ступеня бакалавра

з напряму підготовки
6.050101 “Комп’ютерні науки’’

на тему: Розробка програмного агента моніторингу та управління сонячної електричної станції

Виконав: студент 4 курсу, групи ТМ-51

Задачин Станіслав Сергійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник Ст. викладач Мірошниченко І.В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 "Комп'ютерні науки''

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
"___"_____2019 р.

ЗАВДАННЯ

на дипломну роботу студенту

Задачину Станіславу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи "Розробка програмного агента моніторингу та управління сонячної електричної станції "
керівник роботи Ст. викладач Мірошниченко І.В.
(прізвище, ім'я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від "___"_____ 201__ р. № _____

2. Строк подання студентом роботи ___ _____ 201__ р.

3. Вихідні дані до роботи звіт з моніторингу процесу роботи сонячної електричної станції з інформацією про потенціал генерування електричної енергії

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати існуючі програмні рішення та засоби моніторингу та управління сонячної електричної станції, розробити програмний агент, який у реальному часі може аналізувати та розраховувати потенціал сгенерованої електричної енергії обраної сонячної панелі, кількість спожитої електроенергії за утримання сонячної електричної установки, доцільність її використання за заданих умов.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень) 1. Актуальність 2. Мета та завдання роботи 3. Огляд існуючих рішень 4. Функції системи 5. Етапи розрахунку 6. Використанні програмні засоби 7. Висновки

Дата видачі завдання "___"_____ 201__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2.	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Підготовка матеріалів		
5.	Програмна реалізація системи		
6.	Захист програмного продукту		
7.	Оформлення пояснювальної записки		
8.	Передзахист		
9.	Захист		

Студент _____ Задачин С.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Мірошниченко І.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Пояснювальна записка містить 49 сторінок, включає 21 рисунок, 1 таблицю, 1 формулу та 13 посилань.

Метою дипломної роботи є створення програмного агента управління та моніторингу сонячної електричної станції. Було створено клієнтський додаток за допомогою мови програмування C# у інтегрованій середі розробки Microsoft Visual Studio. Додаток працює з базою даних розробленою за допомогою Microsoft Sql Server.

Розроблено програмний агент моніторингу сонячної електричної станції з функціями моделювання роботи сонячної установки, порівняння сонячних панелей різного типу, засоби збереження розрахованої інформації у файл.

Ключові слова: програмний агент, СЕС, сонячна електрична станція, моделювання, управління, моніторинг, C#, WPF.

ABSTRACT

The explanatory note contains 49 pages, including 21 illustrations, 1 table, 1 formula and 13 references.

The purpose of the thesis is to create a software agent for the management and monitoring of the solar power station. A client application was created using the C# programming language in the Microsoft Visual Studio IDE. The application works with a database developed by Microsoft Sql Server.

The software agent for monitoring the solar power installation with the functions of modeling the solar power station, comparing the various types of solar panels, means of saving the calculated information into a file has been developed.

Keywords: software agent, SPS, solar power station, modeling, management, monitoring, C #, WPF.

АННОТАЦИЯ

Пояснительная записка содержит 49 страниц, включает 21 рисунок, 1 таблицу, 1 формулу и 13 ссылок.

Целью дипломной работы является создание программного агента управления и мониторинга солнечной электрической станции. Было создано клиентское приложение с помощью языка программирования C# в интегрированной среде разработки Microsoft Visual Studio. Приложение работает с базой данных, разработанной с помощью Microsoft Sql Server.

Разработан программный агент мониторинга солнечной электрической станции с функциями моделирования работы солнечной установки, сравнения солнечных батарей различного типа, средствами сохранения рассчитанной информации в файл.

Ключевые слова: программный агент, СЭС, солнечная электростанция, моделирование, управление, мониторинг, C #, WPF.

ЗМІСТ

ВСТУП

В наш час швидкого розвитку альтернативних джерел енергії, особливо енергії сонячного випромінювання, підприємства не можуть обходитися без автоматизованої системи моніторингу сонячної електричної станції, адже вона значно підвищує ефективність роботи, зменшуючи людський фактор. З розвитком альтернативних джерел енергії стає все складніше вести їх облік та розрахунок, а це означає, що необхідна система, що буде автоматично здійснювати розрахунки та допомагати вибирати найбільш ефективну установку.

Під час використання сонячної електричної станції виникають непередбачені заздалегідь чинники, що вимагають оперативного втручання. Отже потрібен функціонал, що забезпечить оперативне внесення змін у існуючі розрахунки та допоможе запобігти прогнозування неактуальних даних.

Одними з найважливіших завдань дипломної роботи є розробка і створення програмного забезпечення для моніторингу та управління сонячних установок енергетичних підприємств, використовуючи сучасні технології та підходи.

Для зручного використання програми було розроблено простий інтерфейс, який дає можливість користувачу швидко отримати всю необхідну інформацію про встановлену сонячну електричну станцію та її роботу.

Програмний продукт надає допомогу ще при виборі сонячних панелей. Можна вказати яку установку користувач хоче придбати, і буде отримано результати роботи такої системи. Також є можливість порівняти різні сонячні панелі та швидко отримати результати найефективнішої системи, що значно полегшує процес вибору.

Для вирішення даної проблеми було поставлено завдання розробити програмний продукт, який би давав змогу підприємцям та власникам приватних сонячних електричних станцій вести розрахунок енергії, що буде вироблено, запобігати виникненню можливих помилок та неполадок в роботі станції, прогнозувати виготовлення електричної енергії, допомагати вибрати найефективніший варіант установки.

Було створено програмний продукт, що має базу даних та в якому користувач має змогу вносити зміни до існуючих установок, автоматично робити їх розрахунок та зберігати його.

Для програмної реалізації було вирішено використати мову програмування C#, технологію Windows Presentation Foundation (WPF) для побудови клієнтських додатків, фреймворк .Net Framework версії 4.6.1. Інтерфейс програмного додатку було створено мовою XAML. Базу даних було розроблено за допомогою Microsoft SQL Server 2017 та Microsoft SQL Server Management Studio 17.

1. ПОСТАНОВКА ЗАДАЧІ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ АГЕНТА МОНІТОРИНГУ ТА УПРАВЛІННЯ СОНЯЧНОЇ ЕЛЕКТРИЧНОЇ СТАНЦІЇ

Метою роботи є створення системи для полегшення рутинної ручної роботи моніторингу роботи сонячної електричної станції, автоматизації процесу розрахунків та побудови графіків, ведення звітності за будь-який період часу роботи установки, процесу вибору найефективнішої установки та прогнозування актуальних даних по виготовленню енергії.

Розроблена система повинна забезпечувати наступні можливості:

— автоматизація процесу розрахунку енергії, що буде виготовлено;

— оптимізація процесу внесення змін в існуючу установку;

— можливість прогнозувати вироблення енергії;

— можливість порівняння та вибору найефективнішої системи сонячних панелей;

На сьогоднішній день існує декілька прикладних програм, які використовуються для моніторингу та управління сонячної електричної станції, але жодна з них не задовольняє повністю необхідним умовам. В дипломній роботі буде продемонстровано використання власного програмного продукту для моніторингу та управління сонячної електричної станції.

Для зручного використання програми було розроблено простий інтерфейс, який дає можливість користувачу швидко отримати всю необхідну інформацію про встановлену сонячну електричну станцію та її роботу.

Програмний продукт надає допомогу ще при виборі сонячних панелей. Можна вказати яку установку користувач хоче придбати, і буде отримано результати роботи такої системи. Також є можливість порівняти різні сонячні панелі та швидко отримати результати найефективнішої системи, що значно полегшує процес вибору.

Програмний засіб розробляється для комп’ютерів під керівництвом системи Windows XP/7/10.

Програмний продукт повинен мати такі функції:

— автоматична генерація звіту на основі розрахунку виробленої енергії;

— функція збереження та дурку звіту;

— розрахунок суми, що буде отримано по зеленому тарифу;

— вибір та порівняння існуючих сонячних установок;

— вивід повного списку характеристик працюючої сонячної електричної станції;

2. ОГЛЯД ІСНУЮЧИХ СИСТЕМ МОНІТОРИНГУ ТА УПРАВЛІННЯ СОНЯЧНОЇ ЕЛЕКТРИЧНОЇ СТАНЦІЇ

Система моніторингу та управління сонячної електричної станції -система, що забезпечує процес контролю, управління і прогнозування вироблення енергії сонячною електричною станцією.

Наразі існують декілька сайтів, що пропонують зробити прогноз виготовлення енергії на основі даних, що вводить користувач, такі як: rentechno.ua, Greenlogic.com, а також atmosfera.ua. Кожний з них має функціонал по вибору області розташування панелей, типу розташування, типу модуля, потужності та площі фотомодулей, азимуту установки, кута нахилу та кількості сонячних панелей. Але жоден з них не спеціалізован саме під вибір сонячних панелей, їх порівняння, прогнозування виробленої енергії та прибуток від використання сонячної електричної станції. Також, це системи, що пропонують користувачу купити сонячні установки, що виробляють саме вони, а не допомагають вибрати їх самому.

У цих системах можна візуалізувати вироблення енергії за допомогою графіків, що відображають дату та вироблену кількість енергії за цей час, але не пропонується відображення виготовлення енергії за поточний день або за певний проміжок часу. У цих системах неможливо порівняти декілька сонячних панелей та вибрати найефективнішу або отримати інформацію про прибуток від продажу електричної енергії. Саме цим і виділяється розроблена система. Також потрібно відмітити, що rentechno.ua, Greenlogic.com, а також atmosfera.ua, це веб ресурси, що не дають користувачам постійного доступу, на відміну від розробленої системи.

Також наведені вище системи не надають функціональності по формуванню звітності роботи сонячної установки за поточний день чи за встановлений проміжок часу.

3. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ

Одним із найважливіших завдань при розробці програмних продуктів є вибір таких засобів програмування, які б полегшили роботу програміста, надавши всі необхідні інструменти для реалізації поставленого завдання, і дали б змогу отримати результат, який повністю задовольняє користувача.

Засоби реалізації, що було використано при створенні програмного продукту наведено на рисунку 3.1.

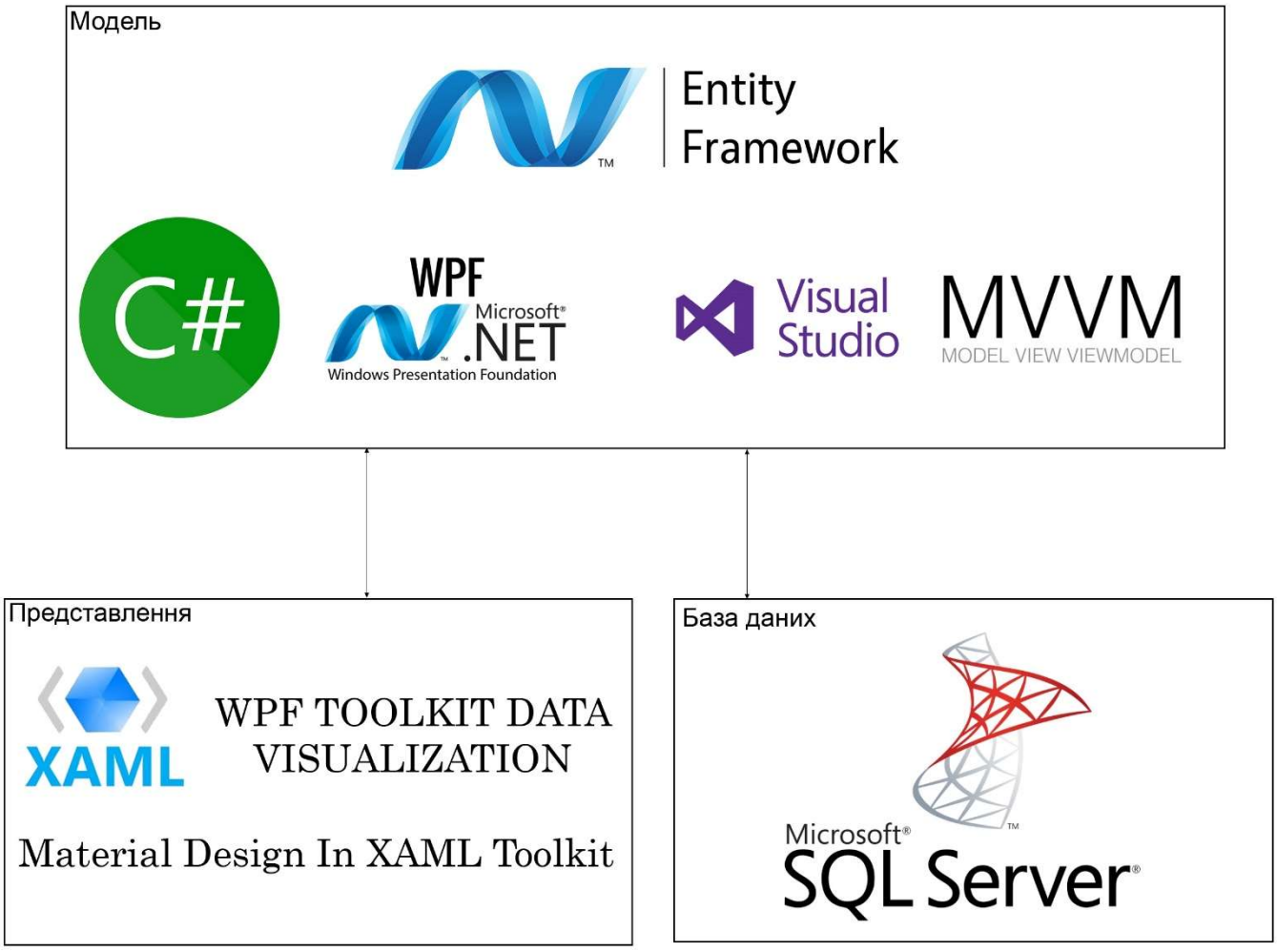

Рисунок 3.1 - Засоби реалізації програмного забезпечення

При створенні програмного забезпечення були використані такі засоби реалізації:

— середовище розробки Microsoft Visual Studio 2017;
— мова програмування C#;
— фреймворк .Net Framework версії 4.6.1;
— фреймворк ADO.NET Entity Framework;
— технологія Windows Presentation Foundation (WPF);
— мова розмітки XAML;
— бібліотека Material Design in XAML Toolkit для розробки графічного інтерфейсу користувача;
— бібліотека WPF Toolkit Data Visualization;
— база даних Microsoft SQL Server 2017 та Microsoft SQL Server Management Studio 17.

Вибір технологій зумовлений тим, що ці інструменти дають найбільшу функціональность серед конкурентів, саме тому були обрані технології - C#, XAML, бібліотеки Material Design in XAML Toolkit та WPF Toolkit Data Visualization. Для розробки бази даних було обрано Microsoft SQL Server 2017.

Розробка програмного продукту обумовлена тим, що здебільшого у фірмах, що займаються виготовленням енергії від сонячних електричних станцій використовується потужна техніка та комп'ютери, а отже застосування програм, що націлені саме на високу продуктивність системи доцільно.

Мовою програмування високого рівня було обрано C#, адже вона має багато бібліотек, що допомагають спростити процес налаштування застосунку, а також стандартизують архітектуру та стиль написання коду, що важливо для можливого розширення функціоналу іншими розробниками.

Також в середовищі розробки Microsoft Visual Studio використовується технологія IntelliSense, що значно прискорює написання коду, покращує роботу з ним та пропонує способи вирішення помилок, що можуть виникнути при розробці.

Базою даних було обрано Microsoft SQL Server 2017 в першу чергу через швидкодію, адже при накопиченні даних це відіграє важливу роль у продуктивності роботи.

3.1 Середовище розробки Microsoft Visual Studio 2017

Середовище розробки Microsoft Visual Studio- це лінійка програмних продуктів фірми Майкрософт, що спеціалізуються на розробці інтегрованого середовища розробки програмного забезпечення. Продукти компанії надають змогу розробляти консольні програми, програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, веб-сайти, веб-застосунки та веб-служби для всіх платформ, що підтримуються компанією. Visual Studio містить редактор вихідного коду з вбудованою технологією IntelliSense, що дуже спрощує процес написання коду, і можливістю швидкого рефакторінга коду без втрати часу. Вбудований відладчик працює одночасно і як відладчик рівня вихідного коду, і як відладчик машинного рівня комп'ютера. Інші інструменти включають в себе редактор форм для поліпшення роботи програміста при розробці графічного інтерфейсу, веб-редактор для інтернет ресурсів, дизайнер класів і дизайнер схеми бази даних.

Середовище розробки Visual Studio надає можливість створювати і додавати до проектів сторонні додатки для поліпшення функціональності на всіх рівнях, включаючи можливість підтримки контролю версій коду, додавання нових наборів інструментів або інструментів для різноманітних аспектів процесу роботи з новими проектами.

Сьогодні сімейство інструментів Visual Studio 2017 містить IDE, сервіс для організації спільної роботи - Visual Studio Team Services, комплексне рішення для реалізації повноцінного циклу розробки мобільних додатків - Visual Studio Mobile Center, мультиплатформенний редактор коду Visual Studio Code, доступний для Mac, Linux і Windows, а також пробна-версія Visual Studio for Mac.

З кожною версією інструментів Microsoft намагається врахувати побажання розробників і зробити їх зручніше для створення додатків практично для будь-якої платформи. Результатом є величезний інтерес і більше 21 млн установок інструменту на сьогоднішній день.

Новітній, полегшений модульний підхід дозволяє розробникам встановити тільки ті компоненти середовища, які їм необхідні і прискорює установку

інструменту від початку і до кінця. До того ж, тепер зникла необхідність створювати проекти і рішення, щоб налагодити будь-який необхідний фрагмент коду.

Середовище розробки Visual Studio допомагає при написанні коду, незалежно від мови, від C #, VB і C ++ до JavaScript і Python, надаючи допомогу в реальному часі.

IntelliSense описує API під час введення, а автоматичне завершення збільшує швидкість і точність роботи. Знайомство з новим API прискорюється завдяки звуженню набору значень за категоріями. Засіб підказки дозволяє перевіряти визначення API. Проблемні місця виділяються знаками тильди, які часто відображаються при введенні.

3.2 Мова програмування C#

Мова програмування C#- це об'єктно-орієнтована мова програмування з безпечною системою типізації на платформі .NET. C# було розроблено Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою центру Microsoft Research.

Синтаксис C# може нагадувати синтаксис таких мов, як C++ і Java. C# має строгу типізацію, підтримуючи поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Багато чого C# перейняв від своїх попередників. У тому числі від мов програмування: C++, Delphi, Модула і Smalltalk. Мова C# виключає деякий список моделей, які були зарекомендовані як проблематичні у використанні при розробці програмних продуктів. Прикладом цього служить відмова корпорації від множинного спадкування класів.

Мова C# розроблялась як мова програмування прикладного рівня для CLR і через це вона залежить, більше всього, від можливостей безпосередньо CLR. Наявність та відсутність деяких особливостей мови програмування обумовлено тим, чи може обрана особливість мови бути трансльована у відповідні конструкції CLR. CLR надає C#, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких

не мають класичні мови програмування. Наприклад, збірку сміття не було реалізовано в самому C#, вона працює завдяки CLR для програм, розроблених на C#.

3.3 Фреймворк .Net Framework 4.6.1

Для створення архітектури систем використовують сучасні бібліотеки, чи фреймворки, які спрощують процес написання коду.

Програмний фреймворк - це комплект готових до використання комплекс програмних рішень, що включають в себе архітектуру побудови проекту, логіку та базову функціональність системи або підсистеми та, навіть, дизайн.

Фреймворк містить в собі набір бібліотек, що пропонують готовий набір рішень, також може містити допоміжні програми, скрипти та загалом все те, що може полегшити роботу програміста при розробці та поєднанні різних компонентів об’ємного програмного забезпечення. Одна з головних переваг використання фреймворків - це стандартизованість структури застосунку.

Фреймворк .Net Framework 4.6.1- крос-платформова технологія, запропонована фірмою Microsoft як платформа для створення звичайних програм та веб-застосунків. Багато в чому технологія являє собою продовження ідей та принципів, закладених ще при розробці технології Java. Однією з головних ідей, покладених в розробку .NET, є сумісність служб, що були написані різними мовами програмування.

Всі бібліотеки в .NET мають дані про свою версію. Це надає змогу позбавитись від можливих конфліктних ситцацій між різними версіями збірок.

Технологія .NET поділяється на дві основні частини - середовище виконання та інструментарій розробки.

Середовища розробки .NET-програм включають в себе використання таких застосунків, як: Visual Studio .NET, SharpDevelop, Borland Developer Studio тощо. Застосовні програми також можна розроблювати в текстовому редакторі та використовувати консольний компілятор.

Середовище розробки .NET розроблює байт-код, що використовується для виконання віртуальною машиною. Вхідна мова такої машини в .NET називається

Common Intermediate Language або CIL, також відома як Microsoft Intermediate Language, або IL. Застосування байт-коду надає можливість отримати крос-платформність саме на рівні скомпільованого проекту, а не- коду, як в мові С. Перед тим як запускається збірки в середовищі виконання байт-код перетворюється вбудованим в середовище JIT-компілятором в машинні коди цільового процесора.

Однією з основних ідей Microsoft .NET є сумісність різних служб, написаних на різних мовах. Наприклад, служба, написана на C ++ для Microsoft .NET, може звернутися до методу класу з бібліотеки, написаної на Delphi; на C# можна написати клас, успадкований від класу, написаного на Visual Basic .NET, а виключення, створене методом, написаним на C#, може бути перехоплено і оброблено в Delphi. На рисунку 3.3 представлено графічний огляд технології .NET.

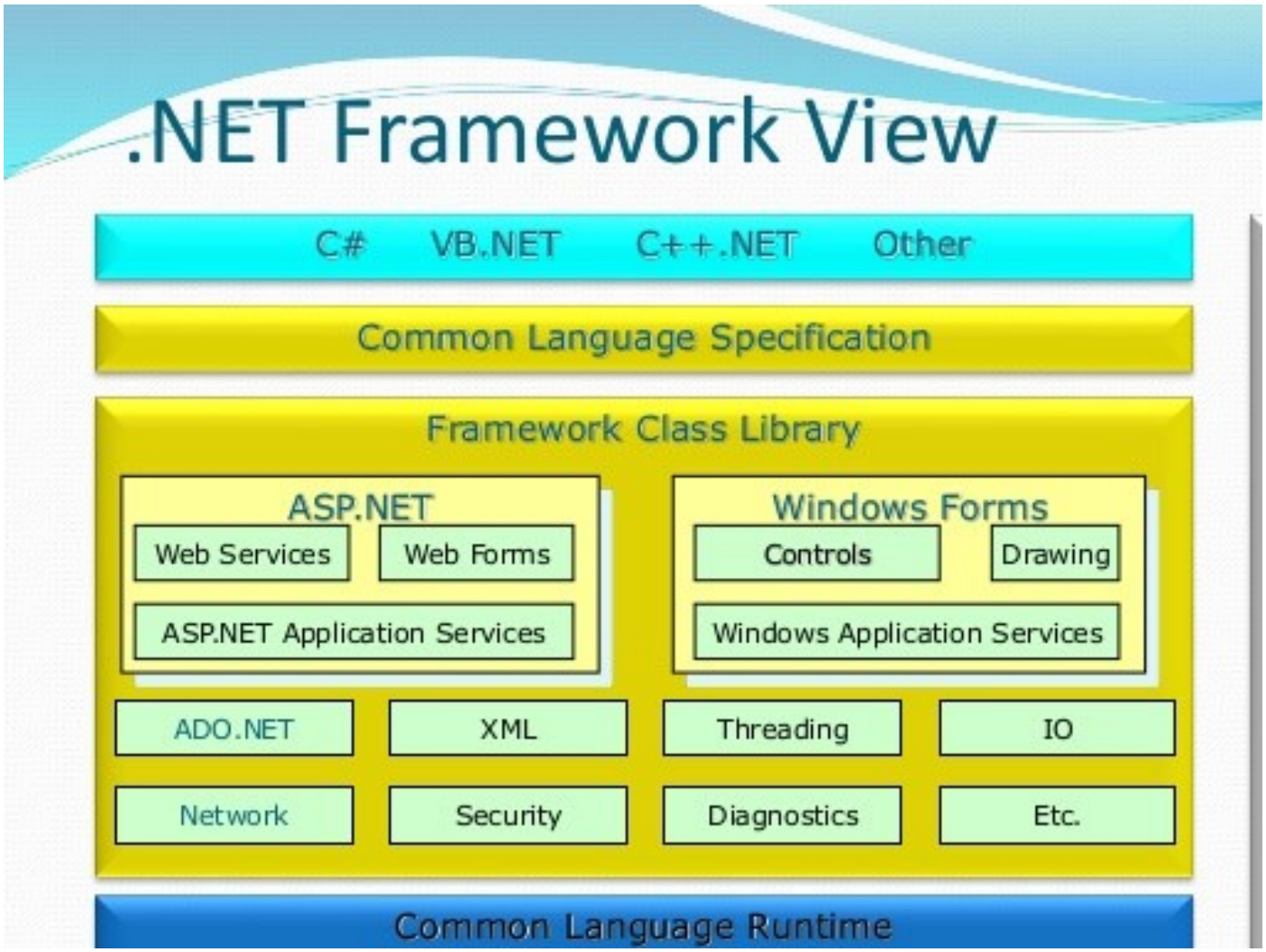

Рисунок 3.2 - Огляд технології .NET

Технологія .NET є патентованою технологією корпорації Microsoft. Проте, після укладення домовленості з компанією Novell, технологія Mono була визнана як реалізація .NET на Unix-подібних системах. Mono надає реалізацію ASP.NET, ADO.NET і Windows.Forms, але в той же час рекомендує обходити ці API.

3.4 Фреймворк ADO.NET Entity Framework

Технологія ADO.NET Entity Framework- об'єктно-орієнтована технологія доступу до даних, є object-relational mapping рішенням для .NET Framework від Microsoft. Надає можливість взаємодії з об'єктами як за допомогою LINQ у вигляді LINQ to Entities, так і з використанням Entity SQL. Для полегшення побудови web-рішень використовується як ADO.NET Data Services (Astoria), так і як зв'язка з Windows Communication Foundation і Windows Presentation Foundation, що дозволяє будувати багаторівневі додатки, реалізуючи один з шаблонів проектування MVC, MVP або MVVM.

Спочатку Entity Framework підтримував підхід Database First, який дозволяв по готовій базі даних згенерувати модель edmx. Потім ця модель використовувалася для підключення до бази даних. Пізніше був доданий підхід Model First. Він дозволяв створити вручну за допомогою візуального редактора модель edmx, і по ній створити базу даних. Починаючи з версії 5.0 більш ефективним підходом став Code First. Його суть закладається в тому, що спочатку пишеться код моделі на C #, а потім по ньому генерується база даних. При цьому модель edmx вже не використовується.

Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework являє собою більш високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. На фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами.

Будь-яка сутність, як і будь-який об'єкт з реального світу, має ряд властивостей. Наприклад, якщо сутність описує людину, то ми можемо виділити такі властивості,

як ім'я, прізвище, зріст, вік, вага тощо. Властивості необов'язково представляють прості дані типу int, а й можуть представляти більш комплексні структури даних. І у всіх сутностей може бути одна або багато властивостей, що відрізняють одну сутність від інших і будуть унікально визначати її. Подібні властивості називають ключами. При цьому суті можуть бути пов'язані асоціативної зв'язком один-до-багатьох, один-до-одного і багато-до-багатьох, подібно до як в реальної бази даних, де відбувається зв'язок через зовнішні ключі.

Відмінною рисою Entity Framework є використання запитів LINQ для вибірки даних з БД. За допомогою Language Integrated Query ми можемо не тільки отримувати певні рядки, що зберігають об'єкти, з бази даних, а й отримувати об'єкти, пов'язані різними асоціативними зв'язками.

Іншим ключовим поняттям є Entity Data Model. Ця модель зіставляє класи сутностей з реальними таблицями в базі даних. Entity Data Model складається з трьох рівнів: концептуального, рівня сховища і рівня зіставлення. На концептуальному рівні відбувається визначення класів сутностей, що будуть використовуватись в додатку. Рівень сховища визначає таблиці, стовпці, відносини між таблицями і типи даних, з якими порівнюється використовувана база даних. Рівень зіставлення служить посередником між попередніми двома рівнями, визначаючи зіставлення між властивостями класу суті і стовпцями таблиць. Таким чином, є можливість через класи, визначені у додатку, взаємодіяти з таблицями з бази даних.

Entity Framework пропонує три можливі способи взаємодії з базою даних:

— Спосіб Database first передбачає створення набору класів за допомогою Entity Framework, які відображають модель конкретної бази даних;
— Спосіб Model first передбачає створення моделі бази даних, за якою потім Entity Framework створе реальну базу даних на сервері;
— Спосіб Code first передбачає створення класів моделі даних, що будуть зберігатися в базі даних, а потім Entity Framework за допомогою цієї моделлю генерує базу даних і її таблиці;

В архітектурі ADO.NET існує прив'язка до фізичної структурі даних, тому при написанні коду для звернення до бази даних необхідно пам'ятати схеми таблиць та

відносин. Для спрощення написання коду і його автоматичної підтримки компанією Microsoft був розроблений Entity Framework, який виводить абстракцію на новий рівень об'єктної моделі. Це дозволило провести відображення структури бази даних на бізнес-об'єкти додатку, в результаті чого дозволило працювати з даними як зі звичайними об'єктами мови. Сутності- це концептуальна модель фізичної бази даних, яка відображається на предметну область. Модель EDM являє собою набір класів клієнтської сторони, які відображаються на фізичну базу даних. Проте, потрібно розуміти, що сутності зовсім не зобов'язані безпосередньо відображатись на схему бази даних, як може здатися, виходячи з назви. Сутнісні класи можна реструктурувати для відповідності існуючим потребам, і виконуюче середовища Entity Framework відобразить ці унікальні імена на коректну схему бази даних. Архітектура Entity Framework відображена на рисунку 3.3.

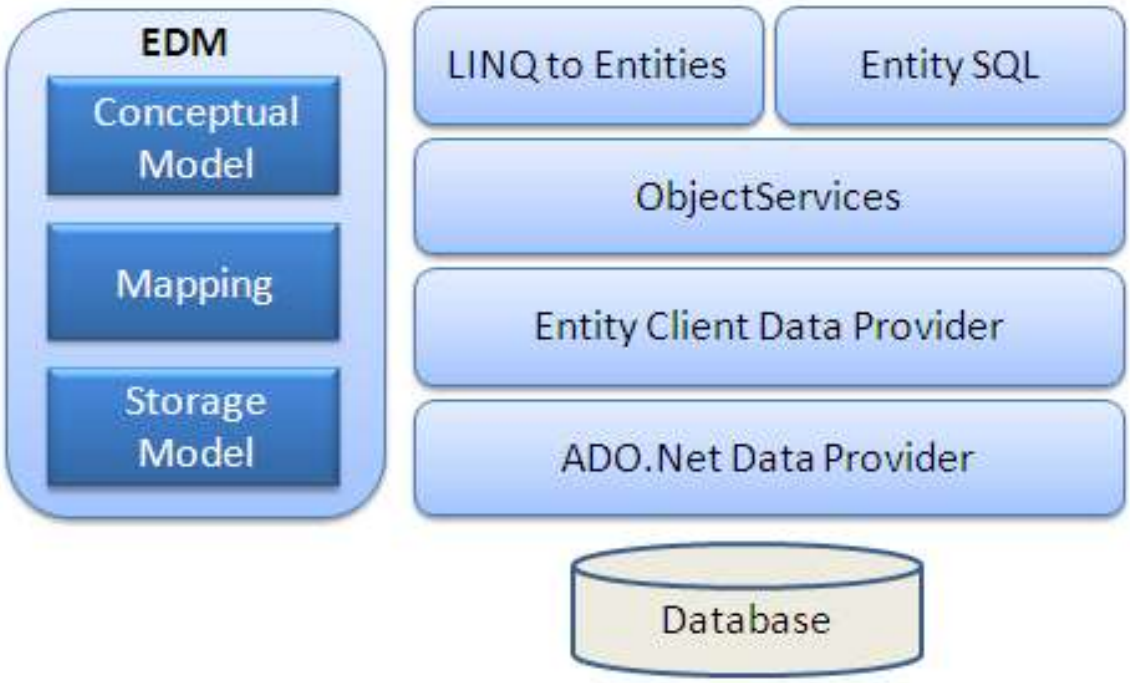

Рисунок 3.3 - Архітектура Entity Framework

Технологія ADO.NET побудована так, щоб ізолювати програміста від вивчення структур баз даних різних виробників, представляючи постачальників баз даних, які інкапсулюють механізм роботи з конкретною СУБД. Це дозволяє створювати адаптери для будь-якої СУБД і повністю використовувати її особливості.

3.5 Система Windows Presentation Foundation

Windows Presentation Foundation- графічна підсистема Microsoft для розробки користувацького інтерфейсу програмних додатків. Технологія має пряме відношення до XAML. WPF разом з .NET Framework 3.0 вбудовано в Windows Vista, також система доступна для установки в Windows.

Це перше оновлення технологічного середовища, що призначене для користувачів інтерфейсу з моменту випуску Windows 95. Воно включає в себе нове ядро для заміни GDI і GDI+, що активно використовувались в Windows Forms. WPF є високорівневим об'єктно-орієнтованим функціональним шаром, що дозволяє створювати двовимірні та тривимірні інтерфейси.

В основу WPF закладено векторну систему візуалізації, що не залежить від розширення пристрою виведення. Вона була створена з урахуванням можливостей сучасного графічного обладнання. WPF дає засоби для створення візуального інтерфейсу, включаючи мову XAML, елементи управління, прив'язку даних, макети, двомірну і тривимірну графіку, анімацію, стилі, шаблони, документи, текст, мультимедію і оформлення.

Графічна технологія DirectX є технологією, що лежить в основі WPF, на відміну від Windows Forms, де використовується GDI / GDI + . Продуктивність WPF вище, ніж у GDI + за рахунок використання апаратного прискорення графіки через DirectX.

Також існує урізана версія CLR, що називається WPF / E, вона ж відома як Silverlight.

Для роботи з WPF потрібна будь-яка .NET-сумісна мова програмуваннння. У цей список входить безліч мов: C #, F #, VB.NET, C ++, Ruby, Python, Delphi, Lua і багато інших. Для повноцінної роботи може бути використана як Visual Studio, так і Expression Blend. Перша орієнтована на програмування, а друга - на дизайн і дозволяє робити багато речей без потреби ручного редагування XAML. Приклади цього - анімація, стилізація, стану, створення елементів управління і так далі.

WPF надає широкий спектр можливостей по створенню інтерактивних настільних додатків. Прив'язка даних- це гнучкий механізм, який дозволяє через

розширення розмітки XAML пов'язувати різні дані від значень властивостей елементів управління до загальнодоступних властивостей, що реалізують поля бази даних через Entity Framework. Прив'язка даних представлена класом Binding, який в свою чергу успадкований від MarkupExtension, що дозволяє використовувати прив'язки не тільки в коді, але і в розмітці.

Крім основного класу Binding в WPF реалізовано ще декілька механізмів прив'язок:

— Механізм MultiBinding дозволяє створювати множинні прив'язки, вказуючи декілька елементів;

— Механізм TemplateBinding використовується в шаблонах для зв'язування властивості елемента всередині шаблону з властивістю елемента, до якого застосовано шаблон;

— Механізм PriorityBinding ранжує список прив'язок і, відповідно до пріоритету, вибирає з них властивість до якої буде застосована прив'язка. Якщо прив'язка, що має найвищий пріоритет успішно повертає значення, то немає необхідності обробляти інші прив'язки в списку;

Основні переваги Windows Presentation Foundation (рисунок 3.4):

— Застосування WPF є незалежним від розширення екрану. Коли створюється додаток в WPF, то цей додаток не залежить від розширення екрану. Він автоматично використовує компоненти DirectX. Роздільна здатність WPF завжди однакова з будь-яким розширенням екрану;

— Замість того, щоб фіксувати елементи керування на місці з певними координатами, WPF підтримує гнучкі потоки, розміщуючи елементи управління на основі їх змісту. В результаті отримується користувальницький інтерфейс, який може бути адаптований для відображення високодинамічного вмісту.

— WPF надає підтримку відтворення будь-якого аудіо або відеофайлу, підтримуємого програмою Windows Media, дозволяючи відтворювати більш одного медіафайлу одночасно. WPF надає в своєму розпорядженні

інструменти для інтеграції відеоспостереження в іншій частині користувальницького інтерфейсу.

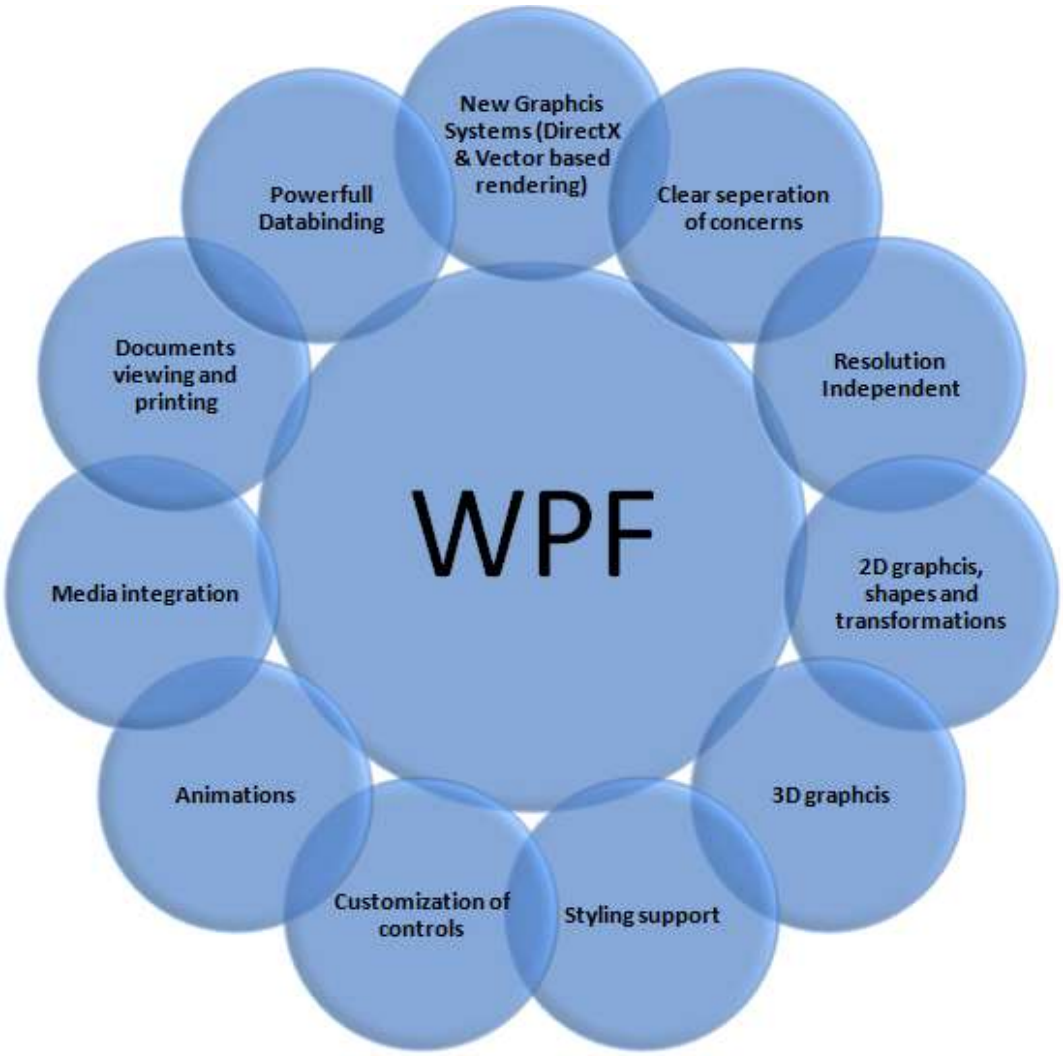

Рисунок 3.4 - Основні переваги Windows Presentation Foundation

WPF в основному використовує декларативну мову XAML. Додаток WPF - це комбінація мов XAML та .NET, таких як C # / VB.Net тощо.

Технологія WPF розбивається на два рівня: managed API і unmanaged API. Managed API містить код, що виконується під управлінням загальномовного середовища виконання .NET- Common Language Runtime. Цей API описує основний функціонал платформи WPF.

Схематична архітектура WPF відображена на рисунку 3.5.

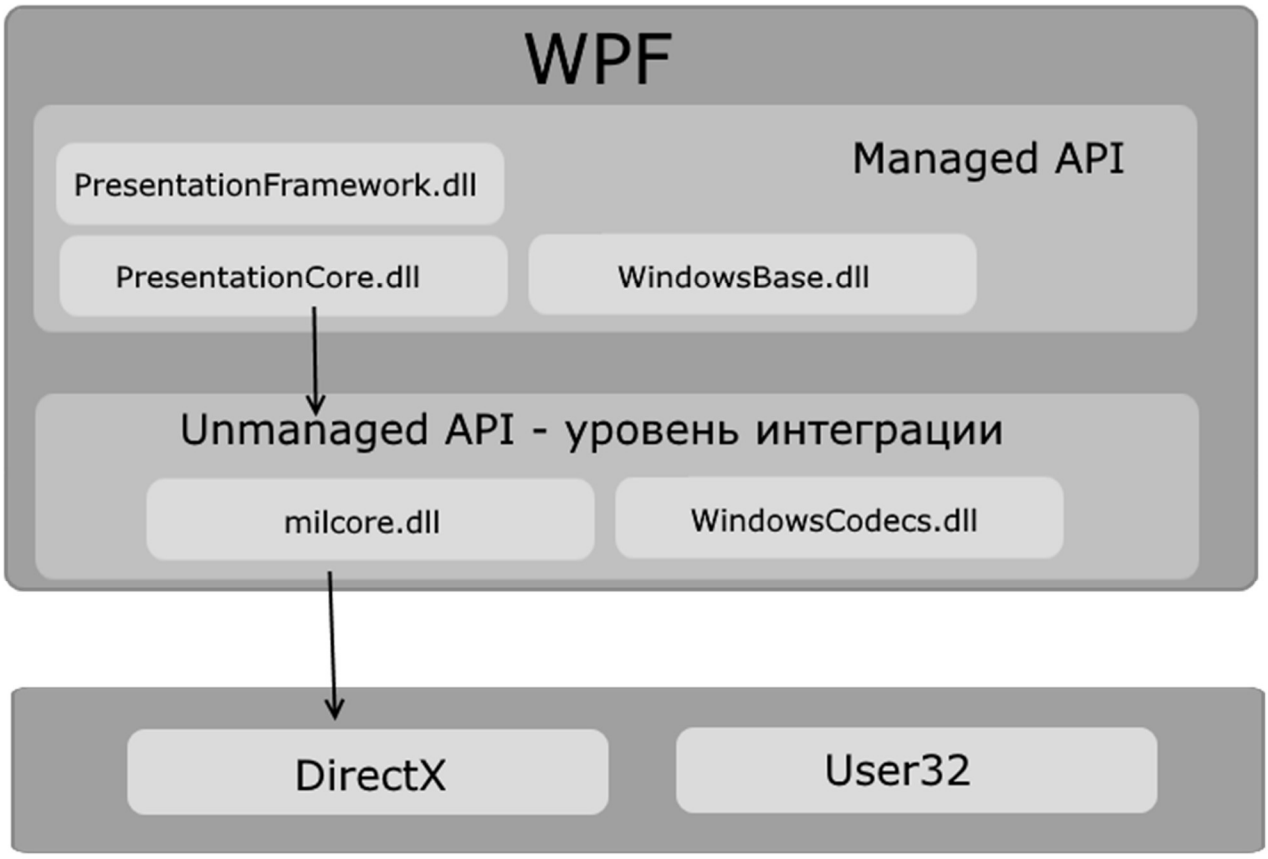

Рисунок 3.5- Схематична архітектура WPF

Windows Presentation Foundation надає розробникам багатий інтерфейс користувача та графіку, яка недоступна у вікнах програми. Як і .NET Framework в цілому, WPF являє собою технологію, орієнтовану на Windows. Це означає, що додатки WPF можуть використовуватися тільки на комп'ютерах, що працюють під управлінням операційної системи Windows. Приклади WPF, основані на браузерах, обмежені аналогічним чином - вони працюють тільки на комп'ютерах Windows, хоча підтримують браузери і Internet Explorer, і Firefox.

WPF є частиною екосистеми .NET і розвивається разом з фреймворком .NET і має ті ж версії. Перша версія WPF 3.0 вийшла разом з .NET 3.0 і операційною системою Windows Vista в 2006 році. З тих пір платформа послідовно розвивається.

3.6 Мова розмітки XAML

Мова eXtensible Application Markup Language є декларативною мовою розмітки. Мова XAML спрощує створення користувацького інтерфейсу для програм для .NET Framework. У розробника є можливість створити елементи інтерфейсу користувача в декларативній розмітці XAML, а потім відокремити визначення користувача інтерфейсу від логіки часу виконання, завдяки використанню файлів коду програмної частини, приєднаних до розмітки за допомогою визначень поділюваних класів. Мова XAML представляє створення екземплярів об'єктів в конкретному наборі резервних типів, що було визначено у збірках. Це і є її відмінністю від більшості інших мов розмітки, які зазвичай, є інтерпретованими мовами без прямого зв'язку з системою резервних типів. Мова XAML надає робочий процес, що дозволяє декільком учасникам розробляти користувацький інтерфейс і логіку програми, при цьому використовуючи потенційно різні засоби.

При поданні у вигляді коду файли XAML являють собою XML-файли, що мають розширення .xaml. Файли можна зберігати у будь-якому кодуванні, що підтримує XML, але, як правило, використовують кодування UTF-8.

Мова розмытки XAML широко використовується в .NET Framework 3.0, особливо в Windows Presentation Foundation (WPF), Windows Workflow Foundation (WWF) і Silverlight. У WPF XAML використовується як мова розмітки для користувача інтерфейсу, для визначення елементів призначеного для користувача інтерфейсу, прив'язки даних, підтримки подій і інших властивостей. У WWF, за допомогою XAML можна визначати послідовності виконуваних дій.

XAML файли можна створювати і редагувати за допомогою інструментів візуального конструювання, таких як: Microsoft Expression Blend, Microsoft Visual Studio, WPF visual designer. Також, їх можна створювати за допомогою стандартного текстового редактора, редактора коду такого як: XAMLPad, або графічного редактора, такого як Vectropy.

Все, що створено або реалізовано в XAML може бути відображено за допомогою більш традиційних .NET мов, таких як: C # або Visual Basic.NET. Однак, ключовим аспектом технології є зменшення складності використовуваних для обробки XAML інструментів, так як XAML заснований на XML. В результаті цього

з'являється безліч продуктів, що створюють засновані на XAML додатки. Оскільки XAML базується на XML, у розробників і дизайнерів існує можливість одночасно працювати над вмістом без необхідності компіляції.

XAML відображає функцію мови, за рахунок чого клас може призначити тільки одне зі своїх властивостей як властивість вмісту XAML. Дочірні елементи даного об'єктного елемента використовуються для завдання значення цієї властивості вмісту. Іншими словами, для властивості вмісту можна опустити елемент властивості, вказавши це властивість в XAML-розмітці, і тим самим створити більш наочну метафору батьківського або дочірнього елементів в розмітці.

3.7 Система Microsoft SQL Server 2017

Система Microsoft SQL Server є комерційною системою керування базами даних, яка розповсюджується компанією Microsoft. Мова, яку використовують для запитів є нічим іншим як Transact-SQL, створеною спільними зусиллями Microsoft та Sybase. Transact-SQL реалізує стандарти ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Мова використовується і для невеликих та середніх за розміром баз даних, і- великих баз даних, що відповідають масштабам підприємств. Багато років система вдало конкурує зі схожими системами керування базами даних.

Базовий код MS SQL Server був написаний на прикладі коду Sybase SQL Server, що дозволило Microsoft вийти на ринок баз даних для підприємств, в якому конкурували Oracle, IBM, так само як і Sybase. Microsoft, Sybase і Ashton-Tate спочатку було об'єднано для розробки і впровадження на ринок першої версії програми, яка отримала назву SQL Server 1.0 для OS/2. Це фактично було еквівалентом Sybase SQL Server 3.0 для Unix та VMS. Microsoft SQL Server 4.2 був випущений у 1992 році та був у складі операційної системи Microsoft OS/2 версії 1.3. Реліз Microsoft SQL Server версії 4.21 для ОС Windows NT було проведено одночасно з релізом самої Windows NT (версії 3.1). Microsoft SQL Server 6.0 був першою версією SQL Server, яку було створено для архітектури NT і без участі в процесі розробки Sybase.

Система Microsoft SQL Server як мову запитів використовує версію SQL, яка отримала назву Transact-SQL, що є програмною реалізацією SQL-92 з багатьма розширеннями. T-SQL надає можливість використовувати додатковий синтаксис процедур, які зберігаються, забезпечує підтримку транзакцій (взаємодія бази даних з керуючим застосунком). Microsoft SQL Server та Sybase ASE для взаємодії з мережею використовують протокол рівня застосунка під назвою Tabular Data Stream.

Система Microsoft SQL Server також надає підтримку для Open Database Connectivity (ODBC)-інтерфейс взаємодії декількох застосунків з СУБД. Версія SQL Server 2005 дохволяє користувачам підключитись через веб-сервер-сервіси, які використовують протокол SOAP, що надає можливість клієнтським програмам, не розробленим для Windows, з'єднуватися з SQL Server. Microsoft також було випущено сертифікований драйвер JDBC, який надає змогу застосункам під керування Java (таким як BEA і IBM Websphere) з'єднуватися з Microsoft SQL Server 2000 і 2005.

Система SQL Server надає можливість дзеркалювати та кластеризувати бази даних. Кластер серверу SQL являє собою сукупність серверів, що було однаково конфігуровано. Ця схема надає можливість розподілити робоче навантаження між декількома серверами. Кожний сервер має одне віртуальне ім'я, а всі дані розподіляються за IP-адресами машин кластеру протягом робочого циклу. У разі відмови або збою на одному з серверів кластеру буде надано автоматичне перенесення навантаження на інший сервер.

Система SQL Server надає модливість надлишкового дублювання даних за трьома можливими сценаріями розробкт:

— Знімок виконує «знімок» бази даних, який сервер потім надсилає користувачам;
— Історія змін розроблена для постійної безпомилковой передачі всіх змін користувачам;
— Синхронізація з іншими серверами передбачає, що бази даних декількох серверів будуть синхронізуватися між один одним. Зміни в усіх базах даних будуть відбуватись незалежно на кожному сервері. А вже у той час коли синхронізація відбувається, системою буде проведена звірка даних.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ

Для автоматизації процесу розрахунку енергії, отриманої від сонячної електричної станції необхідна система, яка буде включати в себе усіх дійових осіб та надавати функціонал для взаємодії між ними, при чому, вся взаємодія має проходити через систему.

Вибір архітектури обумовлений тим, що дана архітектура дозволяє організовувати систему, яка матиме централізоване сховище інформації, що конче необхідно для організації системи моніторингу та управління сонячної електричної станції. Було обрано десктопний додаток, адже це дозволяє взаємодіяти з системою через основний робочий пристрій та незалежно від підключення до інтернету та швидкості передачі даних.

Вибір інструменту для роботи з базою даних обумовлений тим, що такий архітектурний підхід забезпечує гнучкість налаштувань, збільшує швидкість пошуку даних, збільшується надійність зберігання даних, адже зменшується вірогідність несанкціонованої зміни даних.

Для організації архітектури програмного продукту було вирішено використати фреймворк .Net Framework версії 4.6.1, адже такий підхід дозволяє розробити гнучкий програмний продукт з чітко виділеними шарами, реалізацію яких можна змінювати без зміни функціональності застосунку.

4.1. Структура програмного забезпечення

Для реалізації задачі моніторингу та управління сонячної електричної станції був розроблений комп’ютерний додаток для операційної системи Windows. Програмний продукт було реалізовано з використанням фреймворку .Net Framework 4.6.1, це означає, що він має чітко розшаровану архітектуру, де кожен шар може бути замінений. Організацію архітектури проекту показано на рисунку 4.1.

Програмний додаток розроблявся з використанням мови програмування C# платформи .NET Framework 4.0 за шаблоном WPF.

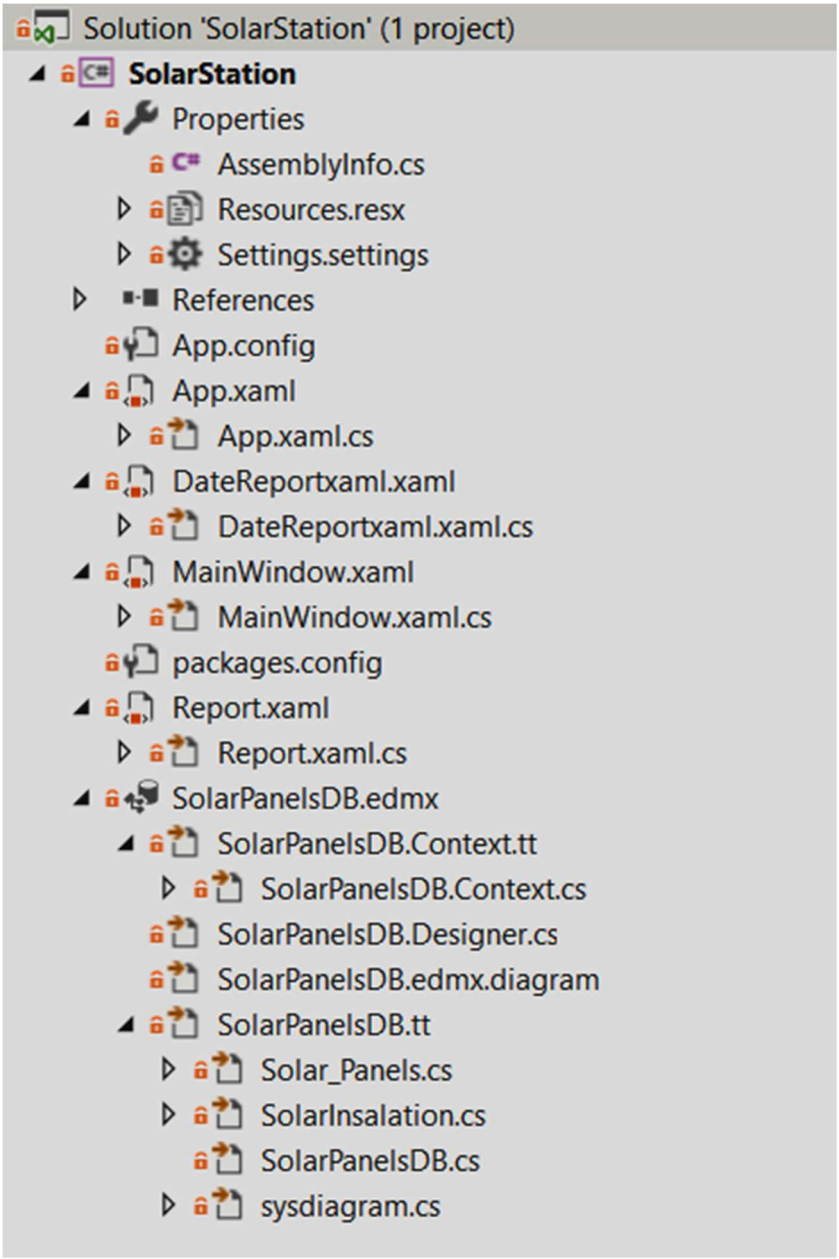

Рисунок 4.1- Організація архітектури проекту

Модель проекту поділяється на декілька класів, кожному з яких відповідає вікно, розроблене за допомогою Windows Presentation Foundation, що і є основним елементом технології.

На рисунку 4.2 відображено структуру організації всіх файлів та папок проекту. Автоматично в середовищі розробки Visual Studio 2017 вони приховані для більш зручної та комфортної роботи з проектом.

Рисунок 4.2- Структура організації всіх файлів проекту

Кожен клас мови C# відповідає за роботу елементів вікон та успадковується від основного класу Window, що дає змогу користуватися основними елементами вікна WPF. Кожен з модулів містить логіку для роботи з базою даних, розрахунку та відображенню даних користувачу. SolarPanelsDB.edmx це модуль сутностей бази даних, розроблений за допомогою ADO.NET Entity Framework, - об'єктно-орієнтована

технологія доступу до даних, є object-relational mapping (ORM) рішенням для .NET Framework. Він надає можливість взаємодії з об'єктами як за допомогою LINQ у вигляді LINQ to Entities, так і з використанням Entity SQL. У результаті такого підключення бази даних Microsoft SQL Server до проекта ми отримуємо таблиці бази даних у вигляді класів, де поля таблиці є полями класу. References - це модуль, що містить набір підключених сторонніх бібліотек, які використовуються проектом.

4.2. Шаблон Model-View-ViewModel

Патерн проектування Model-View-ViewModel- це шаблон проектування, який застосовують під час проектування архітектури програмних додатків. Вперше був представлений Джоном Госсманом у 2005 році. Шаблон MVVM є орієнтованим на сучасні платформи розробки такі, як Windows Presentation Foundation та Silverlight компанії Microsoft.

Шаблон проектування MVVM полегшує відокремлення розробки графічного інтерфейсу від розробки бізнес логіки, відомої як модель. Модель представлення являє собою частину, яка відповідна за перетворення даних для їх подальшої підтримки і використання. Модель представлення більш нагадує модель, ніж представлення і виконує обробку більшої частини логіки відображення даних. Модель представлення також має змогу реалізовувати патерн медіатор, організовуючи доступ до бек-енд логіки навколо множини правил використання, які підтримуються представленням.

Шаблон MVVM використовують для процесу відокремлення моделі та її відображення. Необхідність цього заключається у змозі змінювати їх незалежно одну від одної. Наприклад, розробник буде працювати над логікою роботи з даними, а дизайнер, у цей час, зможе розробляти користувацький інтерфейс.

Патерн MVVM було розроблено з ціллю розподілу роботи дизайнера і розробника програмного забезпечення, що є тяжким, коли Java-розробник намагається побудувати GUI в Swing-javaSC або розробник на Visual C++ створює користувацький інтерфейс в MFC. Зрозуміло, що буде доцільніше, коли дизайнер

інтерфейсів створюватиме інтерфейс, а розробник напише код, який реалізує логіку цього інтерфейсу, але технології типу Swing або MFC не дозволяють бути використаними таким чином. Саме тому доцільно використовувати такий підхід.

Шаблон MVVM доцільно використовувати замість класичного застосування MVC та йому подібних у таких випадках, коли на платформі, де ведеться розробка, присутнє зв'язування даних.

В MVC/MVP зміни у користувацькому інтерфейсі не впливають виключно на модель розробки. Вони йдуть через Контролер. У технологіях, подібних до WPF та Silverlight, наявна концепція зв'язування даних, що дозволяє зв'язувати дані із візуальними елементами в обидва боки.

Архітектура шаблону MVVM вирішує проблеми розподілення обов’язків проектування простим поділом відповідальності:

— Робота з розробки користувацького інтерфейсу здійснюється дизайнером інтерфейсів за допомогою технології, яка більш придатна для такої роботи;
— Логіка користувацького інтерфейсу розробляється програмістом як компонент ViewModel;
— Функціональні зв'язки між користувацьким інтерфейсом та ViewModel реалізовані у системі завдяки зв’язкам, що являють собою певні правила поведінки для відповідей на дії користувача;

Шаблон MVVM можна поділити на такі 3 частини:

— Модель, яка є фундаментальними даними, які необхідні для роботи застосунку;
— Вигляд, який описує графічний інтерфейс для користувача;
— Модель вигляду одночасно виступає абстракцією Вигляду, і надає обгортку даних з Моделі, які мають зв'язуватись.

Графічно архітектуру шаблону MVVM відображено на рисунку 4.3.

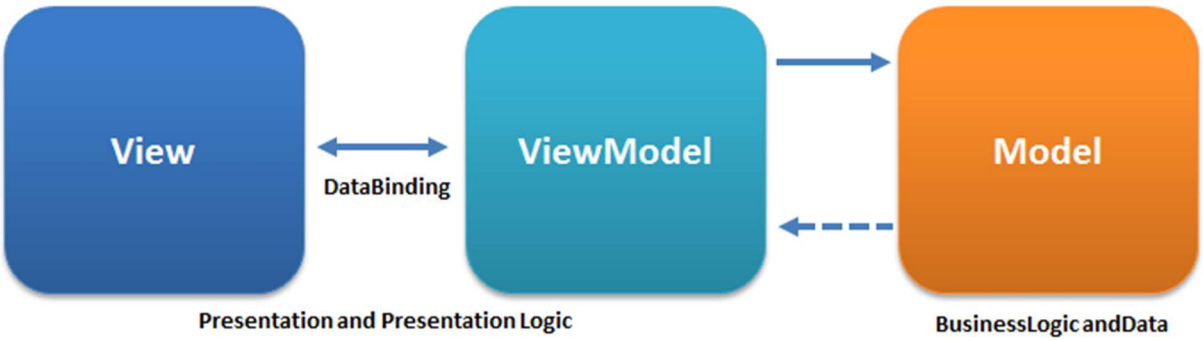

Рисунок 4.3- Архітектура шаблону MVVM

Модель, що була перетворена до Вигляду зберігає у своїй основі команди, що Вигляд використовує для впливу на Модель. Фактично ViewModel призначена для здійснення зв'язку між моделлю та вікном, відслідковування змін даних користувачем та відпрацьовувати логіки роботи View.

4.3. Бібліотека WPF Toolkit Data Visualization

Інструментарій WPF Toolkit Data Visualization- це набір функцій та компонентів WPF, які надаються для розробки програм для .NET Framework. Інструментарій WPF дозволяє користувачам швидше отримувати нові функціональні можливості. Вся функціональність бібліотеки розповсюджується з відкритим вихідним кодом.

Інструментарій був випущений у червні 2009 року і поставляється з набором користувацьких інтерфейсів для візуалізації даних, що називається System.Windows.Controls.DataVisualization.Toolkit.dll, у коді якого розміщено функціональність графіків у WPF.

В дипломній роботі WPF Toolkit Data Visualization було використано для побудови графіків, що відображають виготовлення енергії погодинно та за певний проміжок часу, встановлений користувачем. Приклад роботи бібліотеки відображено на рисунку 4.4.

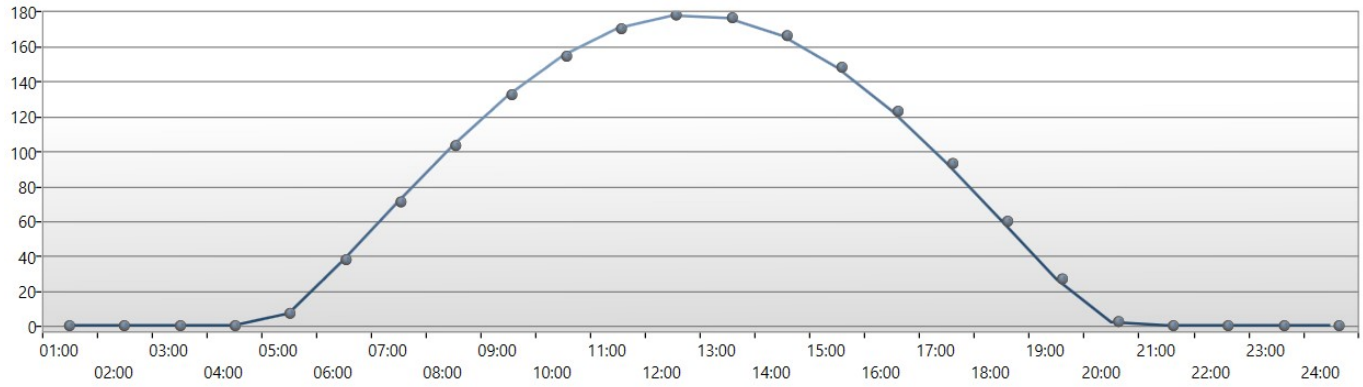

Рисунок 4.4- Приклад використання бібліотеки WPF Toolkit Data Visualization

Інструментарій WPF Toolkit Data Visualization надає широкий спектр можливостей для побудови графіків, які будуть у зручному вигляді надавати користувачу можливість перегляду даних за певний проміжок часу.

При розробці програми можна вибрати найбільш зручний спосіб відображення серед різних типів графіків. У дипломній роботі було обрано для використання графік типу LineSeries, який є найбільш підходящим для поставленої задачі.

4.3. Бібліотека Material Design In XAML Toolkit

Бібліотека Material Design In XAML являє собою набір інструментів для швидкого та зручного надання програмному додатку більш сучасного вигляду за допомогою автоматичної зміни елементів у розмітці XAML.

Завдяки бібліотеці Material Design In XAML розробник може легко принести красиві програмні додатки до життя, використовуючи сучасну та популярну мову дизайну. Приклад використання бібліотеки наведено на рисунку 4.5.

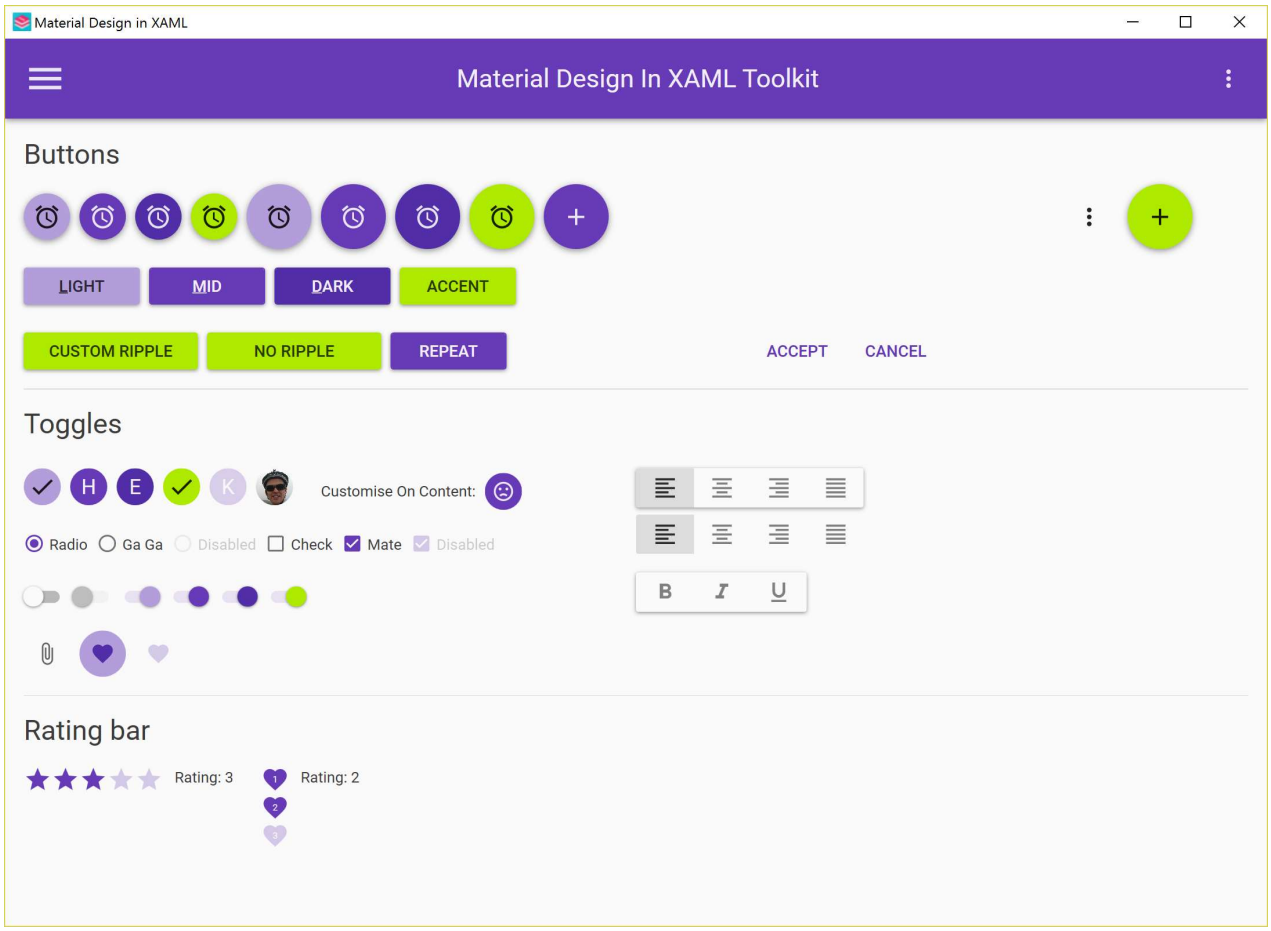

Рисунок 4.5- Приклад використання бібліотеки Material Design In XAML

Робота з бібліотекою значно прискорює роботу програміста при написанні коду, адже позбавляє його потреби у розробці власного стилю програмного додатку та налагоджування зовнішнього вигляду.

5. МАТЕМАТИЧНИЙ ПІДХІД ДО ВИРІШЕННЯ ЗАДАЧІ

При розробці програмного продукту важливим чинником є розуміння математичної частини. Для вирішення задачі моніторингу сонячної елетричної станції та розрахунку виготовлення енергії було використано наступну формулу:

$$E = \frac{N \cdot I \cdot V \cdot Ko \cdot K\text{вит.}}{24}, \quad (5.1)$$

де E – реальна потужність батареї; N– кількість панелей, що працюють; I– кількість сонячної енергії, що отримують панелі під час роботи; V– номінальна потужність панелі; Ko – поправочний коефіцієнт кількості сонячної енергії, що падає на панель; Квит – коефіцієнт втрати електричної енергії в системі автономного електропостачання.

Результати розрахунків програмного продукту точно відображають кількість енергії, що було виготовлено сонячною установко.

6. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для роботи системи необхідно встановити програмний додаток на комп’ютер з операційною системою Windows. Користувач потрапляє на стартову сторінку з формою для вибору сонячної установки та інших характеристик. Далі в залежності від вибраної установки користувач може почати процес отримання результатів роботи сонячної електричної станції, перегляду повних характеристик поточної установки та порівнянню декількох панелей.

6.1. Системні вимоги

Для коректної роботи з розробленою програмною системою користувачу потрібні мінімальні потужності апаратного забезпечення. Вимоги наведені в таблиці 6.1.

Таблиця 6.1.- Вимоги до апаратного забезпечення користувача

Пристрій	Характеристика
Процесор	Intel ® Core ™ 2 / 2 Duo / Pentium ® / Celeron ® / Xeon™ / i3 / i5 / i7 чи AMD 6 / Turion ™ / Athlon ™ / Duron ™ / Sempron ™ з тактовою частотою не нижче 1.5 GHz.
Оперативна пам’ять (RAM − Random Access Memory)	Рекомендовано не менше 2Гб RAM

Підтримувані апаратні архітектури:

— 32-розрядна (x86);
— 64-розрядна (x64)

6.2. Сценарії роботи користувача з програмним продуктом

Перше, що бачить користувач при роботі з програмою, це вікно вибору характеристик сонячної панелі. За замовчуванням програма надає користувачу ті налаштування системи, які він використовує, але є змога корегувати їх або вибрати інші, якщо користувач хоче дізнатися про роботу системи з іншою установко. Характеристики установки, доступні для змінення відображено на рисунку 6.1.

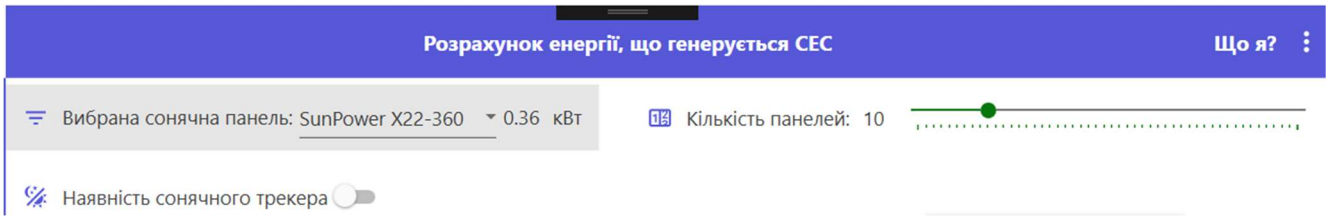

Рисунок 6.1- Характеристики за замочуванням сонячної установки

Далі користувач може вибрати встановлену сонячну панель або ту панель, яку він хоче протестувати. Мається на увазі отримати результати розрахунків виготовлення енергії у реальному часі установкою, яку він вибере вручну.

У цьому вікні користувач має такі можливості, як: вибір сонячної панелі, вибір кількості встановлених панелей, вибір наявності інстальованого сонячного трекера, що служить для виготовлення максимально можливої кількості енергії сонячними панелями. При зміні кожної з цих характеристик буде змінюватися і графік, за допомогою якого дуже зручно можна побачити кількість енергії, що буда отримана за поточну годину. Для більш точного відображення розрахунків користувач може навести курсор миші на один з показників. Головне вікно програмного додатку відображено на рисунку 6.2.

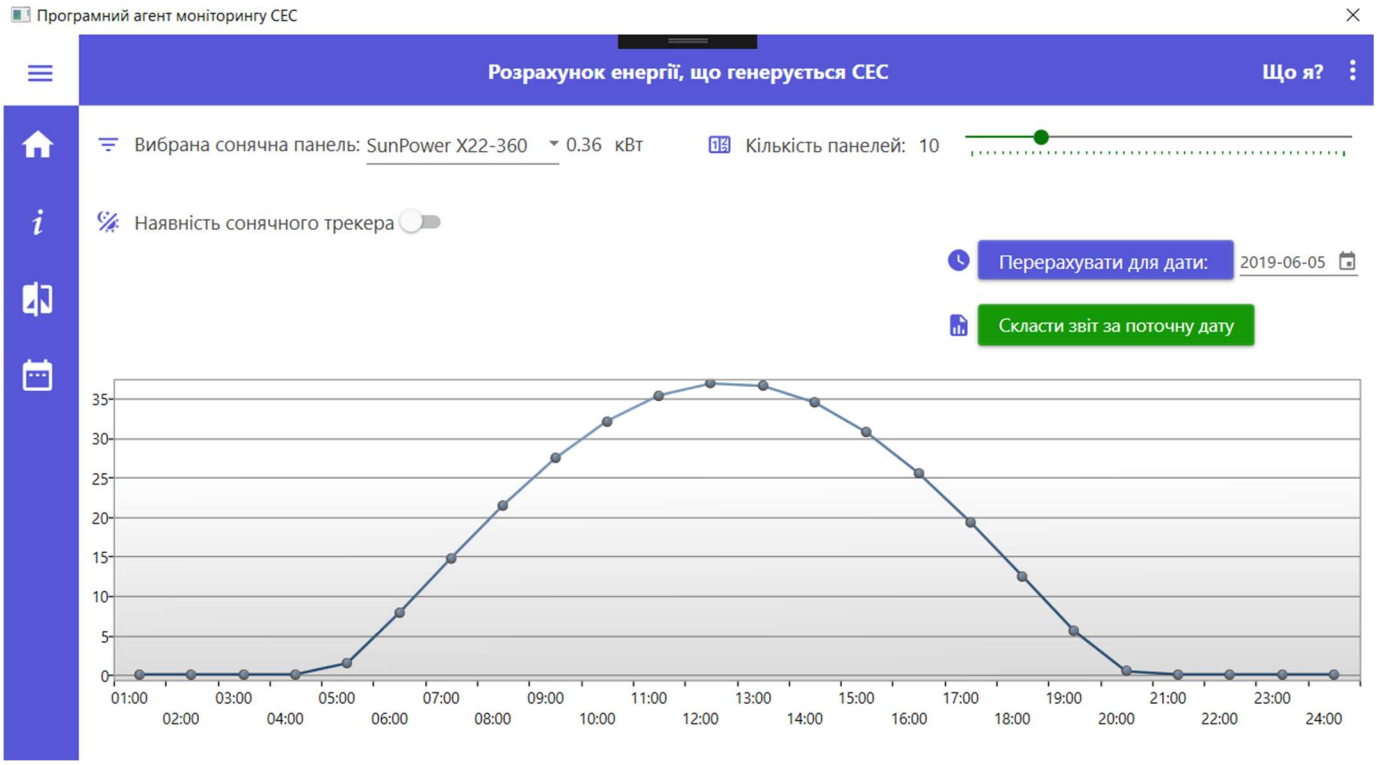

Рисунок 6.2- Головне вікно програмного додатку

На горизонтальній осі графіку відображено кожна година дня, а вертикальна показує скільки кіловат енергії за годину буде виготовлено.

При зміні кожної з наведених характеристик автоматично буде змінюватись і графік. Якщо користувач, наприклад, захоче змінти кількість сонячних панелей, що використовується для виготовлення енергії і увімкнути сонячний трекер, йому лише необхідно змінити положення повзунка і переключити кнопку відповідно. Графік буде побудовано автоматично, у користувача немає необхідності кожний раз натискати на зайві кнопки, щоб переглянути зміни. Такий підхід є дуже зручним та зберігає час.

На головному вікні користувач також має змогу отримати розрахунки з виготовлення енергії за будь-яку іншу дату. Для цього необхідно обрати час, за який користувач хоче отримати дані та натиснути кнопку “Перерахувати для дат”. Результат розрахунку для іншої дати наведено на рисунку 6.3.

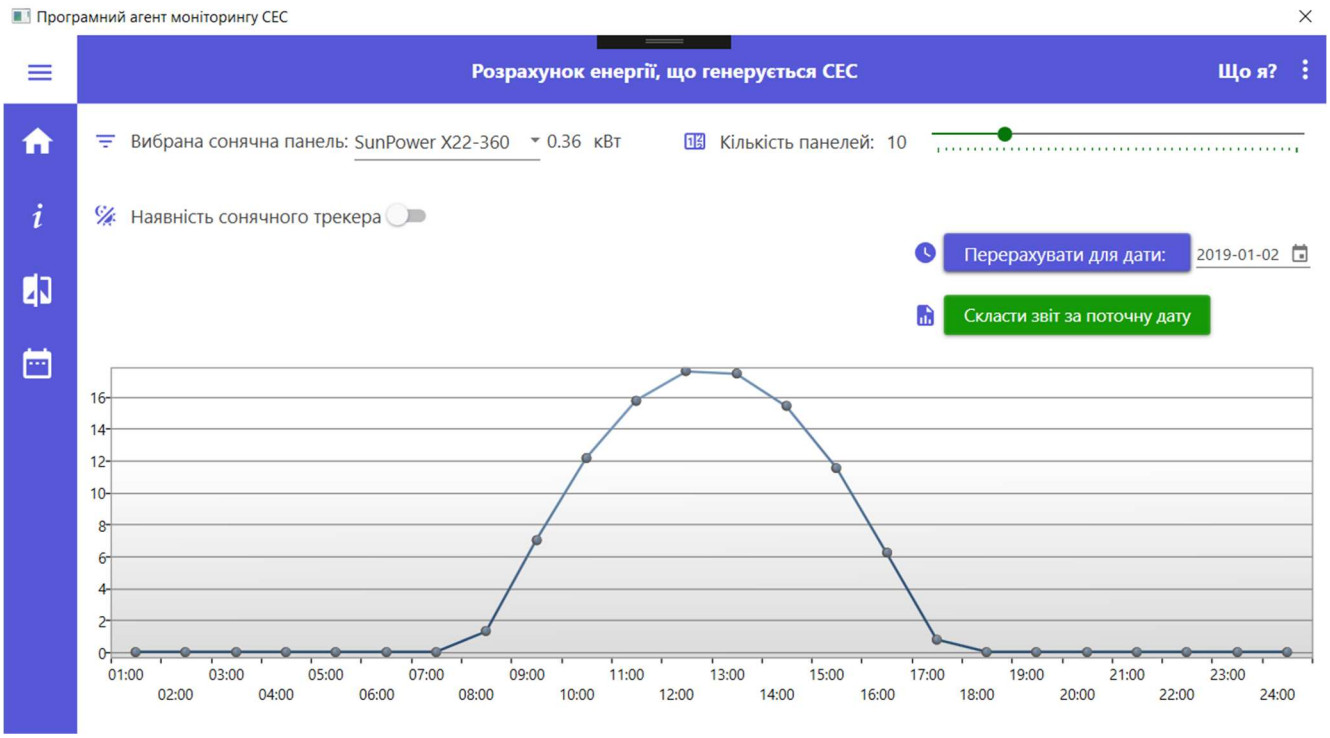

Рисунок 6.3- Розрахунок енергії для іншої дати

Для зручного перегляду результатів виготовлення енергії та їх зберігання було розроблено систему звітування. Для отримання звіту з результатами роботи сонячної електричної станції користувач повинен натиснути кнопку “Скласти звіт га поточну дату”.

У вікні звіту користувач отримує інформацію про установку, яку він використовує або тестує, яка включає в себе назву панелі та її номінальну потужність. Також звіт відображає кількість панелей, які працювали в цей день та наявність працюючого сонячного трекера. Звіт також надасть інформацію про суму коштів, яку було витрачено на придбання сонячних панелей.

Якщо користувача цікавить інформація про прибуток від встановлення сонячних панелей, йому буде надано суму коштів, яку буде отримано від продажу енергії за зеленим тарифом.

Також система надасть користувачу інформацію про сумарну кількість енергії, яку було виготовлено за поточний день та відобразить графік виготовлення енергії погодинно. Вікно звіту відображено на рисунку 6.4.

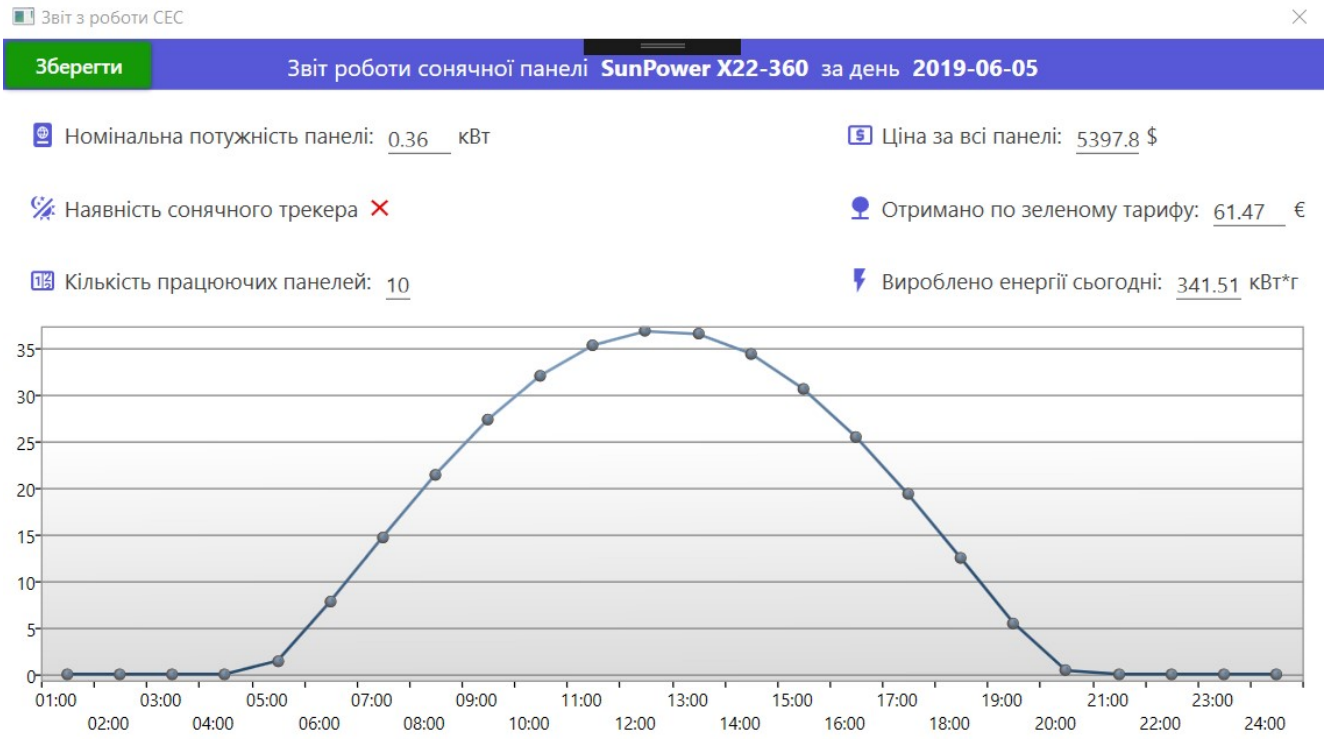

Рисунок 6.4- Вікно звіту

Також користувач має змогу зберегти та роздрукувати звіт, якщо йому необхідна ця інформація для ведення статистики роботи сонячної електричної станції. Приклад звіту у форматі pdf наведено на рисунку 6.5.

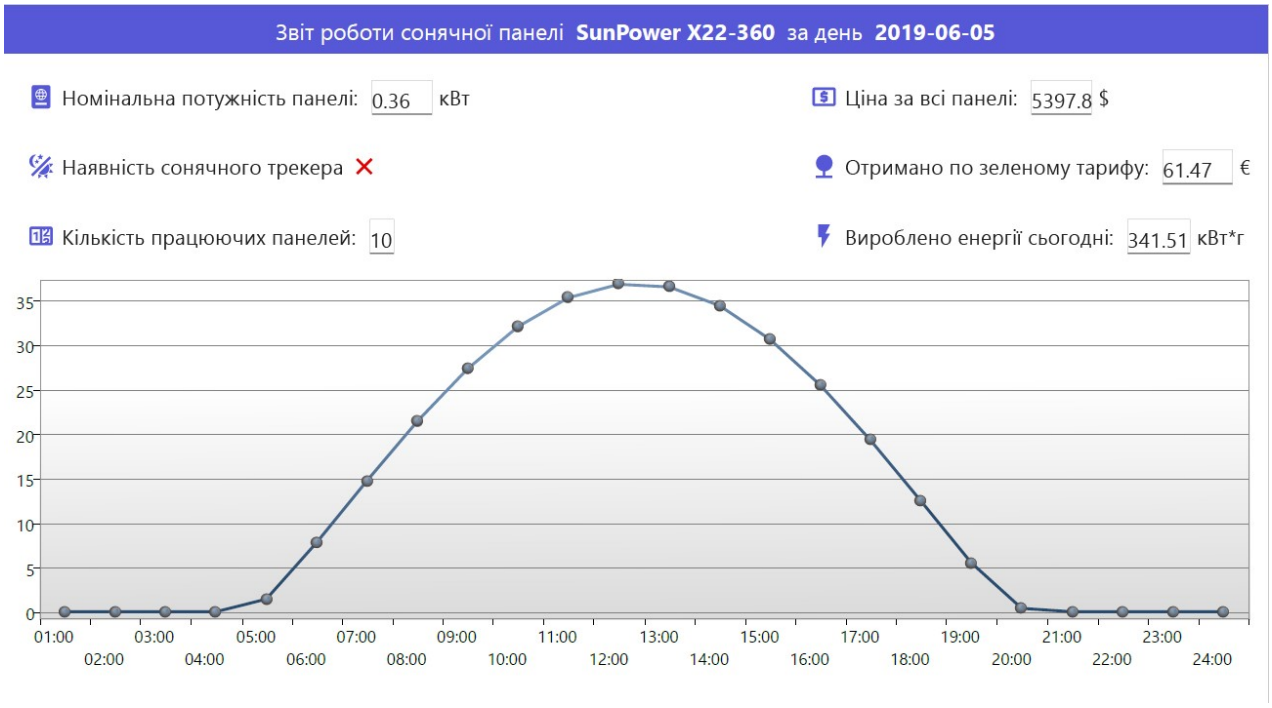

Рисунок 6.5- Збережений звіт у форматі pdf

Для навігації у системі та переходу між вікнами було розроблено панель зліва, завдяки якій користувач може швидко перейти до наступного вікна, як це показано на рисунку 6.4.

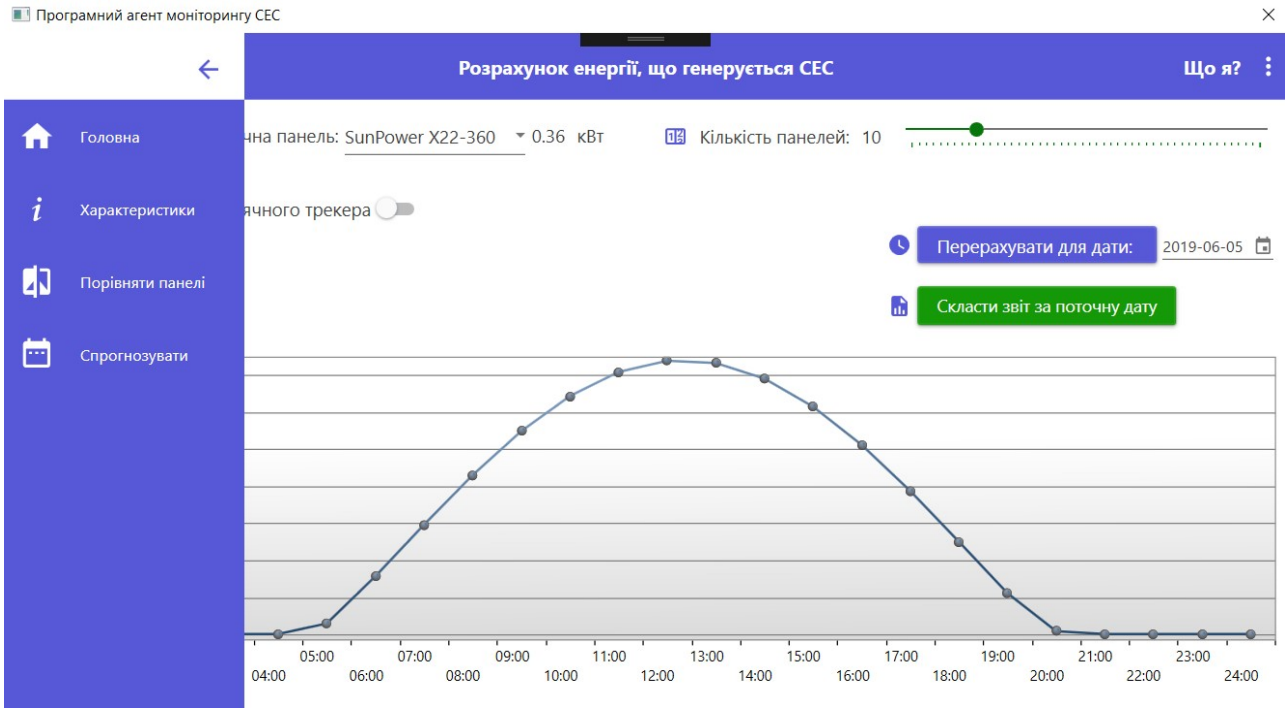

Рисунок 6.4- Панель навігації

Користувачем такої системи може бути кожен, хто використовує сонячну електричну станцію. Немає значення чи це домашня СЕС або встановлена на підприємстві, що використовує енергію сонячного випромінювання.

Саме тому користувачу надано можливість отримати необхідні характеристики сонячних панелей, які він використовує та які є у базі даних. Вікно характеристик сонячних панелей відображено на рисунку 6.5.

За замовчуванням в цьому вікні користувачу буде надано інформацію про панелі, які він використовує, але є можливість переглянути технічні характеристики панелей, які наведені в базі даних. Для цього необхідно вибрати панель зі списку, і програма автоматично відобразить характеристики такої установки. У вікні наведено тільки ті характеристики, які можуть бути цікавими для користувача без зайвої інформації.

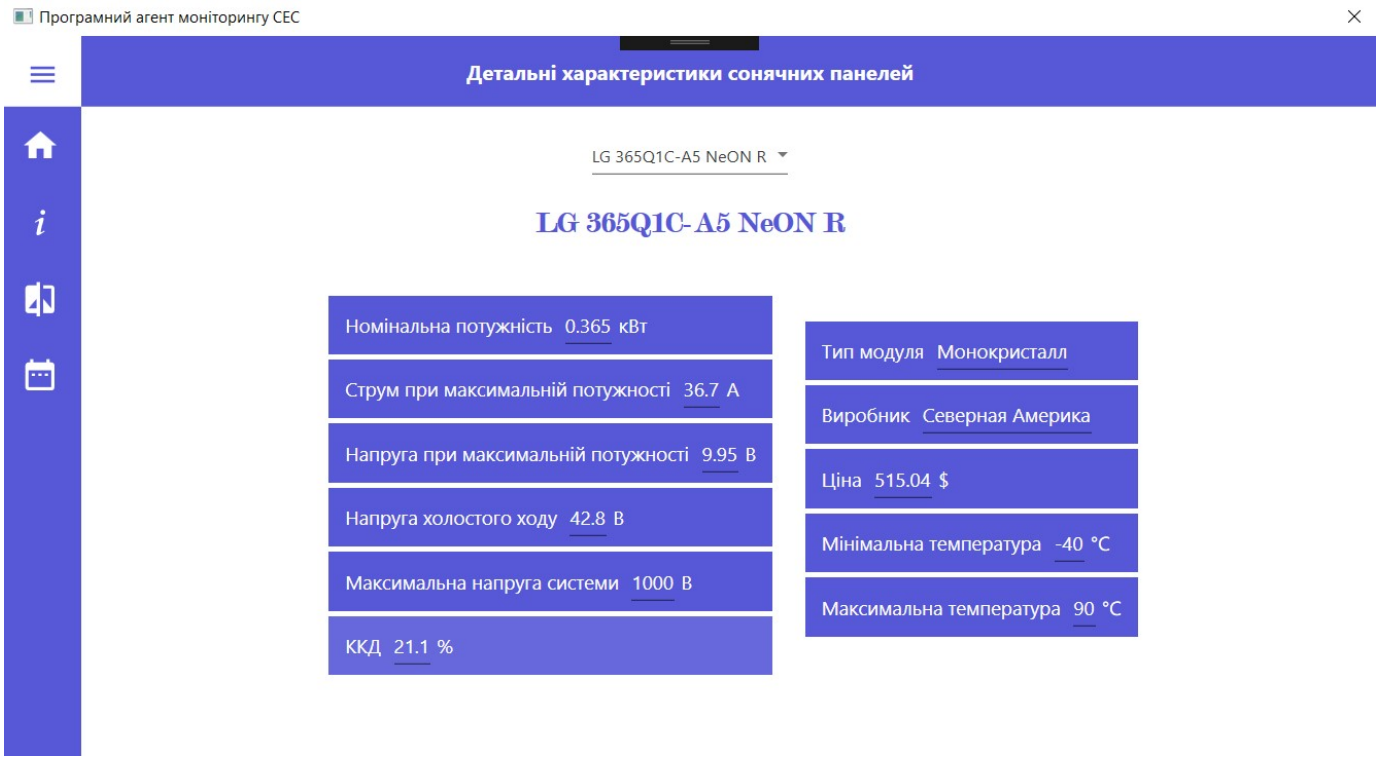

Рисунок 6.5- Вікно характеристик сонячних панелей

Розроблений програмний продукт надає не тільки результати розрахунку енергії, але й допомагає користувачу ще при виборі сонячних панелей. Для цього користувач може скористатися вікном порівняння сонячних панелей.

На вікні порівняння користувач має змогу порівняти і обрати найефективнішу установку. Для цього йому необхідно вибрати сонячну панель зі списку та натиснути на кнопку “Порівняти”. У результаті користувач отримає два списки таких характеристик, як: номінальна потужність сонячної панелі, ціну за одну панель, її коефіцієнт корисної дії, максимальна напруга сонячної панелі, тип модуля та країна виробник.

Для більш комфортного користування програмою всі показники панелей, що порівнюються підсвічені різними кольорами, де зелений означає більш ефективний результат, а червоний- гірший. Вікно порівняння сонячних панелей наведено на рисунку 6.6.

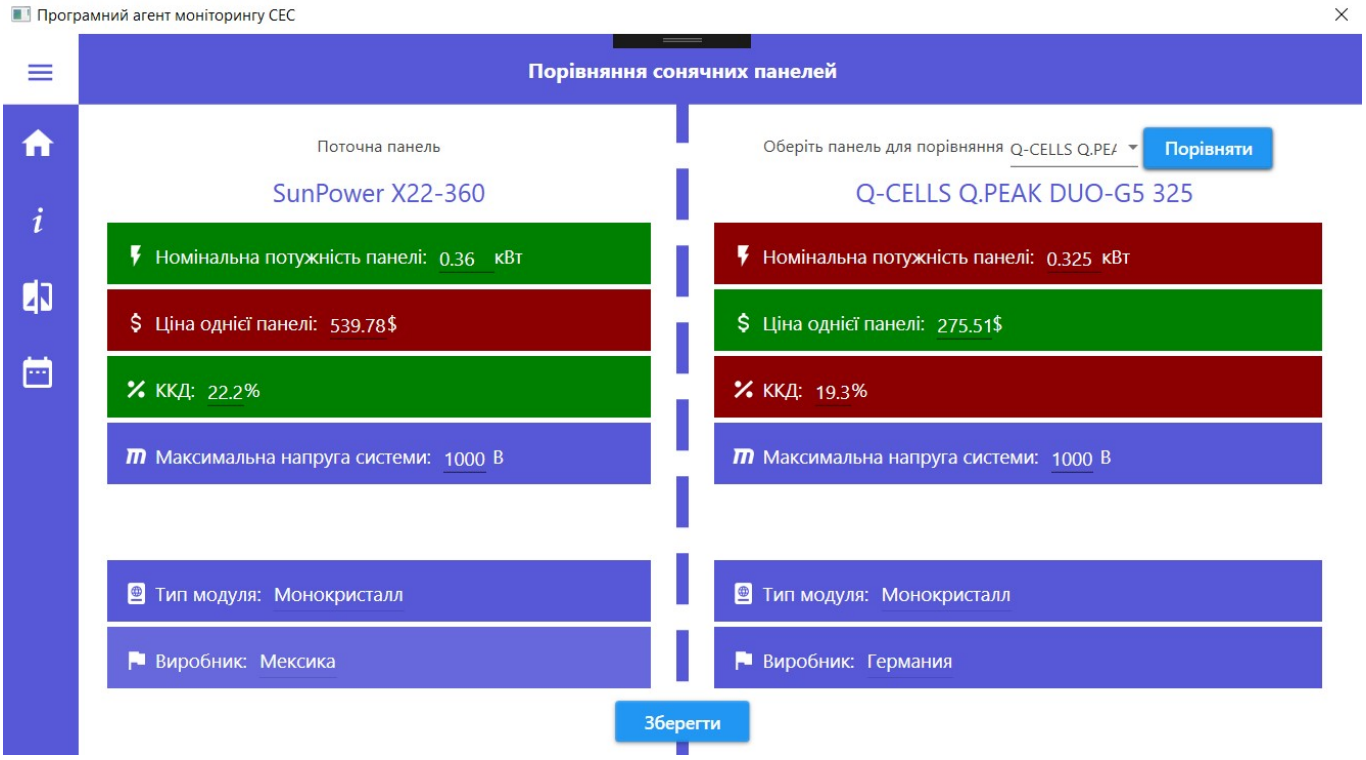

Рисунок 6.6- Вікно порівняння сонячних панелей

Звіт з порівнянням панелей можна зберегти або роздрукувати, натиснувши відповідну кнопку знизу вікна. Приклад звіту наведено на рисунку 6.7.

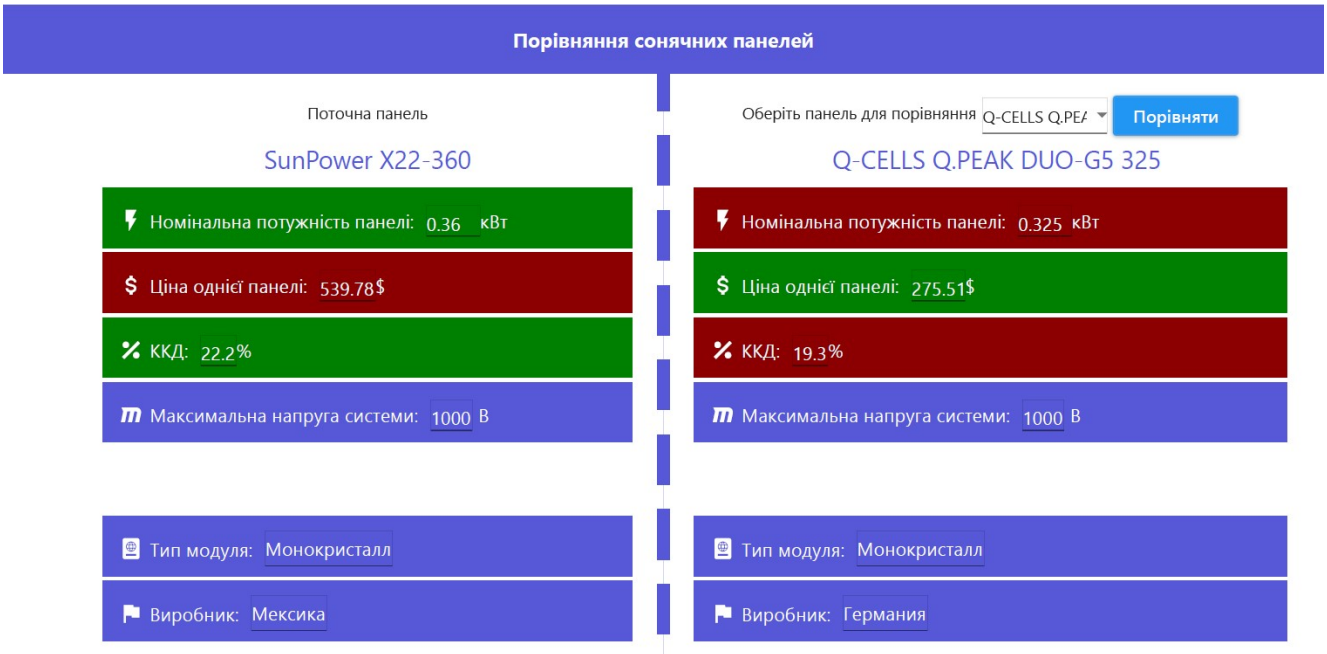

Рисунок 6.7- Звіт порівняння сонячних панелей

При роботі з сонячною електричною станцією користувачу знадобиться інформація про виготовлення енергії не тільки за сьогоднішній день, але й за певний проміжок часу. Для цього було розроблено систему, яка дає користувачу можливість вибрати певний проміжок часу і отримувати результати розрахунків енергії за цей час. Вікно розрахунку енергії за проміжок часу відображено на рисунку 6.8.

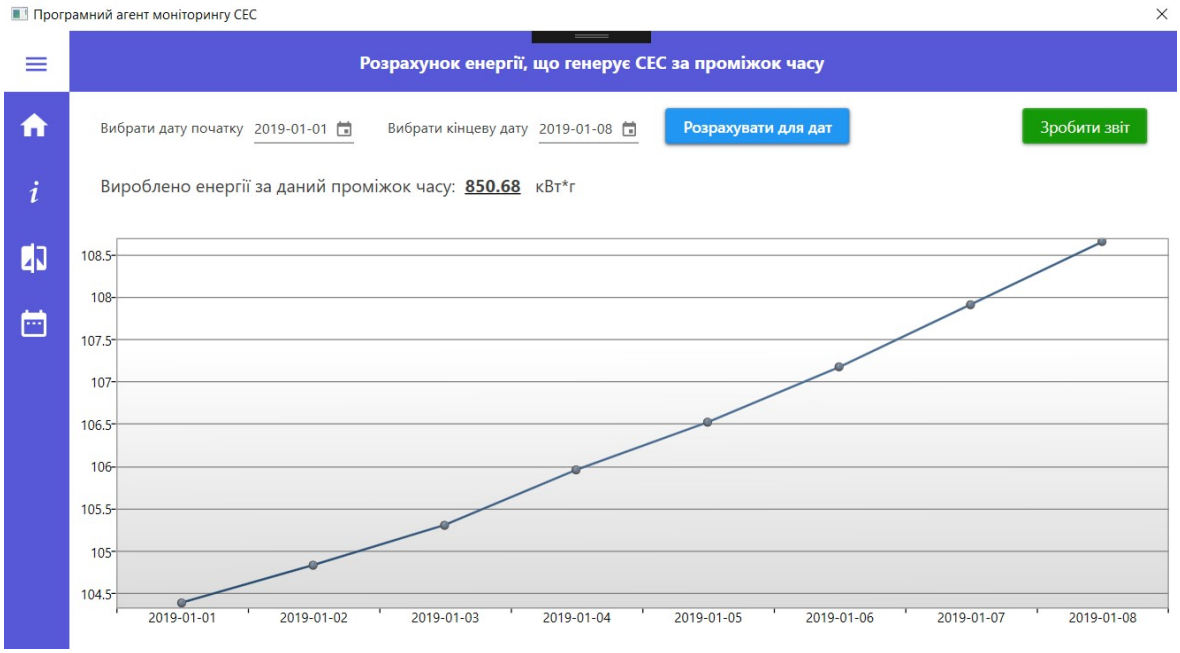

Рисунок 6.8- Вікно розрахунку енергії за вказаний проміжок часу

Для отримання результатів з виготовлення енергії за проміжок часу користувач повинен вибрати дату початку розрахунку та дату закінчення. Після натиснення кнопки “Розрахувати для дат” користувачу буде надано графік з результатами розрахунку енергії, де горизонтально відображено кожен день з заданого проміжку часу, а вертикально відображається кількість виготовленої енергії. Лінії на графіку показують скільки енергії було отримано за конкретний день. Сумарна кількість енергії, що було виготовлено за проміжок часу буде відображено для користувача.

Також система надає можливість прогнозувати виготовлення енергії за деякий проміжок часу. Користувач повинен розуміти, що цей розрахунок є не точним і розроблений на основі приблизних значень сонячної інсоляції. Користувач системи має змогу отримати звіт з роботи сонячної електричної станції за проміжок часу.

Такий звіт можна буде зберегти та роздрукувати для більш комфортної роботи. Приклад звіту з роботи станції наведено на рисунку 6.9.

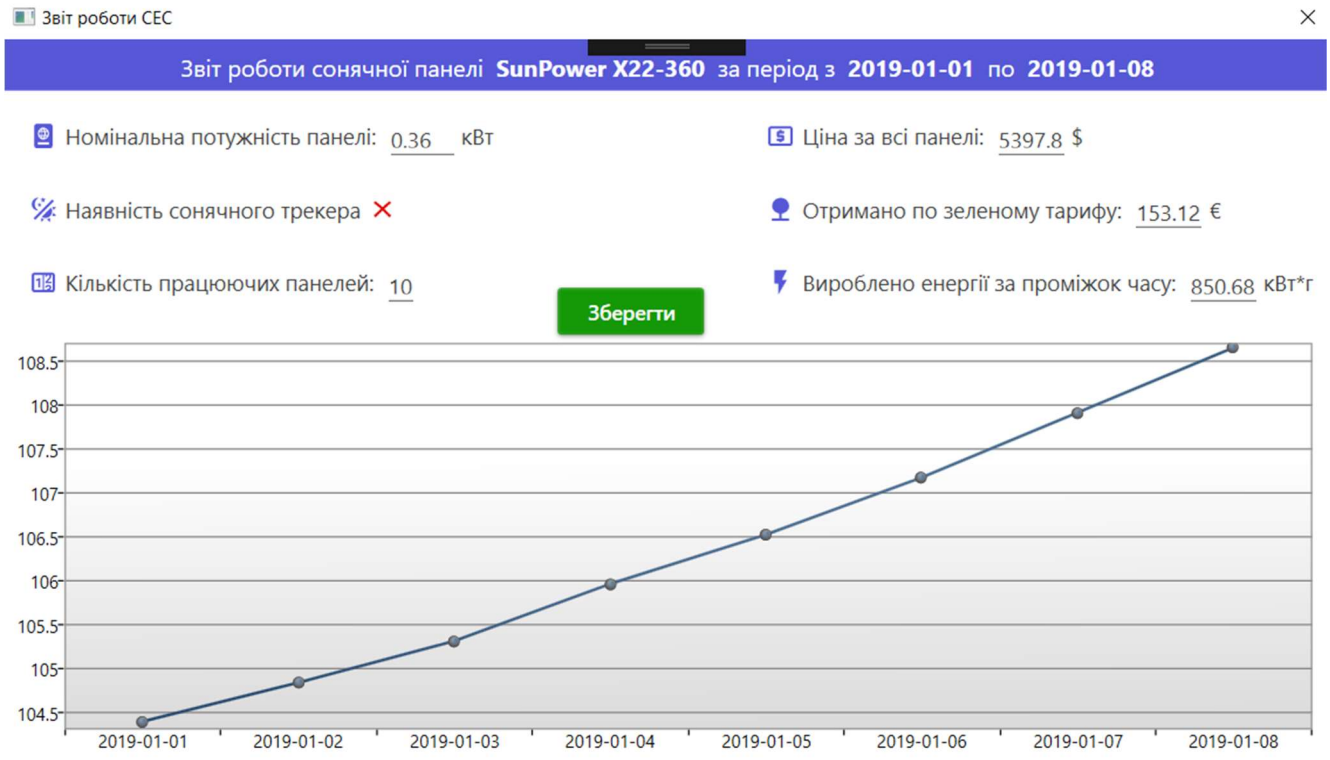

Рисунок 6.9- Звіт з розрахунку енергії за вказаний проміжок часу

У вікні звіту користувач отримує інформацію про установку, яку він використовує або тестує, яка включає в себе назву панелі та її номінальну потужність. Також звіт відображає кількість панелей, які працювали в цей перыод часу та наявність працюючого сонячного трекера. Звіт також надасть інформацію про суму коштів, яку було витрачено на придбання сонячних панелей.

Якщо користувача цікавить інформація про прибуток від встановлення сонячних панелей, йому буде надано суму коштів, яку буде отримано від продажу енергії за зеленим тарифом. Також система надасть користувачу інформацію про сумарну кількість енергії, яку було виготовлено за обраний промыжок часу та відобразить графік виготовлення енергії. Звіт з роботи сонячної електричної станції можна зберегти та роздрукувати для подальшої роботи.

ВИСНОВКИ

У ході виконання даної роботи було розроблено програмний продукт для моніторингу роботи сонячної електричної станції.

Десктопний додаток було розроблено мовою C# з використанням WPF.

Програмний продукт може бути використано для поєднання з системою виготовлення енергії альтернативних джерел.

Для демонстрації роботи розробленого продукту було створено демонстраційний додаток, в якому відображається робота розробленої системи, виводяться графіки результатів виготовлення енергії, та надається можливість зберігати результати розрахунків.

Користувач має змогу власноруч задавати усі параметри системи сонячної електричної станції. Для всіх параметрів системи генерується графік з результатами розрахунку енергії.

Результати, які отримує користувач, мають невелику похибку, що враховується при виведенні розрахунків. Дана система істотно спрощує процес розрахунку енергії сонячною електричною станцією та дозволяє досягти чіткого результату.

У ході аналізу існуючого програмного забезпечення моніторингу генерування енергії сонячною електричною станцією було досліджено системи, які слугують для вирішення поставлених задач. Аналіз показав, що існуючі системи вирішують задачу не у повному обсязі та мають ряд недоліків.

Було проведено огляд методів і засобів розробки програмного додатку та обґрунтовано вибір створення програмної системи, а також побудованої архітектури. Це дає змогу підвищити гнучкість та зручність системи, як у розробці та супроводі, так і у використані.

За результатами виконання тестових завдань підтверджена коректність отриманих результатів, отже система відповідає поставленим вимогам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Адам Фримен - ASP.NET Core MVC с примерами на C# для профессионалов. 2017р.

2. Джеймс Чамберс - ASP.NET Core. Разработка приложений. 2018р.

3. Markus Egger - MVVM Survival Guide for Enterprise Architectures in Silverlight and WPF.

4. Martin Fowler - GUI Architectures. Часть 1. 2009р.

5. Martin Fowler - GUI Architectures. Часть 2. 2009р.

6. Болье А. - Learning SQL. 2005р.

7. Альтернативные источники энергии и энергосбережение. Германович В., Турилин А. Санкт-Петербург: Наука и Техника, 2014. 320 с.

8. Рихтер Д. CLR via C# Программирование на платформе Microsoft .NET Framework 2.0 на языке C# / Джеффри Рихтер. – Киев: Питер, 2008. – 656 с.

9. Троелсен Э. Язык программирования C# и платформа .NET 4.5 / Эндрю Троелсен. – Киев: вильямс, 2014. – 1312 с.

10. Бодягін І. Model-View-Controller в .Net / Іван Бодягін. // RSDN Magazine. – 2016. – №2.

11. Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Э.Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – Харьков: Питер, 2001. – 368 с.

12. Скіт Д. C# in Depth / Джон Скіт., 2013. – 582 с.

13. Lampropoulos, I. (2014). Energy management of distributed resources in power systems operations Eindhoven: Technische Universiteit Eindhoven. -167c.

Додаток 1

Програмний агент управління та моніторингу сонячної електричної станції

Специфікація

УКР.НТУУ“КПІ”.ТМ51107_19Б

Аркушів 2

2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ ім. Ігоря Сікорського».ТМ51107_19Б 81-1	Записка	Пояснювальна записка
Компоненти		
УКР.НТУУ«КПІ». ТМ51107_19Б 12-1	Текст програмного модулю	
УКР.НТУУ«КПІ». ТМ51107_19Б 13-1	Опис програми	

Додаток 2

Програмний агент управління та моніторингу сонячної електричної станції

Текст програмного модулю

УКР.НТУУ“КПІ”.ТМ51107_19Б 12-1

Аркушів 7

2019

```
namespace SolarStation
{
    public partial class MainWindow : Window
    {
        double sumCompare = 0;
        public int panelAmount;
        public Solar_Panels SolarPanelSelected;
        public Solar_Panels SolarPanelSelectedInf;
        public Solar_Panels SolarPanelSelectedCompareItem;
        public SolarPanelEntities sp = new SolarPanelEntities();
        public List<Solar_Panels> SolPal { get; set; }
        public List<Solar_Panels> SolPalInf { get; set; }

        public MainWindow()
        {
            InitializeComponent();
            FillAbout();
            DatePick.SelectedDate = DateTime.Now;
            startDate.SelectedDate = DateTime.Now;
            BindCB();
            BindCBInfPanel();
            SolarPanelListCB.ItemsSource = sp.Solar_Panels.ToList();
            PanelAmountSl.ValueChanged += Slider_ValueChanged;
            PanelAmountSl.Value = 10;
            isTrackSun.Checked += CheckBox_Checked;
            isTrackSun.Unchecked += CheckBox_Unchecked;
        }

        #region Main
        private void BindCB()
        {
            var item = sp.Solar_Panels.ToList();
            SolPal = item;
```

```
            DataContext = SolPal;
        }


        public void ComboBox_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            SolarPanelSelected = SolarPanelListCB.SelectedItem as
Solar_Panels;
            double kWh = (double)SolarPanelSelected.NominalPower_W /
1000;
            V.Text = kWh.ToString();
            FillChart();


            compareThis.Text =
SolarPanelSelected.NamePanel.ToString();
            PowerComparable.Text = kWh.ToString();
            PriceComparable.Text =
SolarPanelSelected.Price__.ToString();
            EffComparable.Text =
SolarPanelSelected.PanelEfficiency.ToString();
            maxPowComparable.Text =
SolarPanelSelected.MaxSystemVoltage_V.ToString();
            CrystallComparable.Text =
SolarPanelSelected.SolarCells.ToString();
            manufacturerComparable.Text =
SolarPanelSelected.Manufacturer.ToString();
        }
        public void FillChart()
        {
            Style styleLegend = new Style { TargetType =
typeof(Control) };
            styleLegend.Setters.Add(new Setter(Control.WidthProperty,
0d));
            styleLegend.Setters.Add(new Setter(Control.HeightProperty,
0d));
            Chart.LegendStyle = styleLegend;
```

```
            List<KeyValuePair<string, double>> KeyValue = new
List<KeyValuePair<string, double>>();
            DateTime date = DatePick.SelectedDate.Value;
            double perDayPower = 0;


            if (date.Year != 2019)
            {
                int years = 2019 - date.Year;
                date = date.AddYears(years);
            }
            if (isTrackSun.IsChecked == true)
            {
                foreach (var time in sp.SolarInsalations.Where(x =>
x.Date == date).ToList())
                {
                    double power =
SolarPanelSelected.CalculatePower((int)time.ETRN, panelAmount);
                    perDayPower += power;
                    KeyValue.Add(new KeyValuePair<string,
double>(time.Time, power));
                }
                ((LineSeries)Chart.Series[0]).ItemsSource = KeyValue;
            }
            else
            {
                foreach (var time in sp.SolarInsalations.Where(x =>
x.Date == date).ToList())
                {
                    double power =
SolarPanelSelected.CalculatePower((int)time.ETR, panelAmount);
                    perDayPower += power;
                    KeyValue.Add(new KeyValuePair<string,
double>(time.Time, power));
                }
                ((LineSeries)Chart.Series[0]).ItemsSource = KeyValue;
            }
```

```
            day.Text = perDayPower.ToString("#.##");



        }

        private void Slider_ValueChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            ((Slider)sender).SelectionEnd = e.NewValue;
            this.panelAmount = (int)PanelAmountSl.Value;
            FillChart();
        }

        private void CheckBox_Unchecked(object sender, RoutedEventArgs
e)
        {
            FillChart();
        }
        private void CheckBox_Checked(object sender, RoutedEventArgs
e)
        {
            FillChart();
        }



        private void OpenReport_Click(object sender, RoutedEventArgs
e)
        {
            Report reportWindow = new Report();
            reportWindow.Show();
        }
        #endregion

        #region Information Tab
        private void BindCBInfPanel()
```

```
        {
            var itemInf = sp.Solar_Panels.ToList();
            SolPalInf = itemInf;
            DataContext = SolPalInf;
        }

        public void ComboBox_SelectionInfChanged(object sender,
SelectionChangedEventArgs e)
        {
            SolarPanelSelectedInf = SolarPanelListInfCB.SelectedItem
as Solar_Panels;
            double kWh = (double)SolarPanelSelected.NominalPower_W /
1000;
            ShowInfo();
            //V.Text = kWh.ToString();
            //FillChart();
        }

        public void ShowInfo()
        {
            Solar_Panels itemInf = SolarPanelListInfCB.SelectedItem as
Solar_Panels;
            //PanelNameTxt.Text = item.NamePanel.ToString();

            Header.Content = itemInf.NamePanel.ToString();

            double kWh = (double)itemInf.NominalPower_W / 1000;
            NominalPowTxt.Text = kWh.ToString();

            RatedVoltageTxt.Text = itemInf.RatedVoltage_V.ToString();

            RatedCurrentTxt.Text = itemInf.RatedCurrent_A.ToString();

            OpenCircuitVoltageTxt.Text =
itemInf.OpenCircuitVoltage_V.ToString();
```

```
            MaxSystemVoltageTxt.Text =
itemInf.MaxSystemVoltage_V.ToString();


            PanelEfficiencyTxt.Text =
itemInf.PanelEfficiency.ToString();


            SolarCellsTxt.Text = itemInf.SolarCells.ToString();


            ManufacturerTxt.Text = itemInf.Manufacturer.ToString();


            PriceTxt.Text = itemInf.Price__.ToString();


            MinTempTxt.Text = itemInf.MinTemperature.ToString();


            MaxTempTxt.Text = itemInf.MaxTemperature.ToString();
        }
        #endregion
        private void CalculateBtn_Click(object sender, RoutedEventArgs
e)
        {
            FillChart();
        }


        #region Compare
        private void SolarPanelCompareList_SelectionChanged(object
sender, SelectionChangedEventArgs e)
        {
            //SolarPanelSelectedCompareItem =
SolarPanelCompareList.SelectedItem as Solar_Panels;
            //compareTo.Text =
SolarPanelSelectedCompareItem.NamePanel.ToString();
            //double kWh =
(double)SolarPanelSelectedCompareItem.NominalPower_W / 1000;
            //PowerComparaTo.Text = kWh.ToString();
```

Додаток 3

Програмний агент управління та моніторингу теплонасосної установки

Опис програмного модулю

УКР.НТУУ“КПІ”.ТМ51107_19Б 13-1

Аркушів 6

2019

АНОТАЦІЯ

Розроблений програмний агент моніторингу та управління сонячної електричної станції має функції моделювання процесу генерації електричної енергії сонячною установкою впродовж довільного проміжку часу, функції збереження звіту процесу моделювання, можливість переглядати збережений звіт, функції конфігурування об’єкту моніторингу.

Користувачами можуть бути люди, які мають комп'ютер з операційною системою Windows.

ЗМІСТ

1. ВІДОМОСТІ ПРО ПРОГРАМНИЙ МОДУЛЬ

При створенні програмного забезпечення були використані такі засоби реалізації:

— середовище розробки Visual Studio 2017, адже має найбільше функціоналу серед конкурентів;

— мову програмування C#;

— фреймворк .Net Framework для організації архітектури та побудови програмного додатку мовою C#;

— мову розмітки XAML;

— Фреймворк ADO .Net Entity Framework для роботи з побудованою базою даних;

— Бібліотеку WPF Data Visualization для розробки графіків виводу інформації;

— Бібліотеку WPF Material Design Data Visualization для розробки графічного інтерфейсу користувача;

— Git для впровадження системи контролю версій;

1.1. Опис логічної структури

Для реалізації задачі моніторингу та управління сонячної електричної станції у складі мультиагентної системи був розроблений програмний додаток, який може бути розгорнутий у будь-якому комп’ютері під операційною системою Windows.

Інтерфейс користувача було розроблено з використанням таких технологій: XAML мова розмітки додатків, WPF система для побудови клієнтських додатків, C# мови програмування.

1.2. Вхідні та вихідні дані

Вхідними даними для системи є дані сонячної інсоляції України за період 2016-2019 років та дані користувача що характеризують сонячну установку.

Вихідними даними є звіт з моделювання роботи сонячної електричної станції, що несе у собі інформацію про потенціал сгенерованої енергії, кількість коштів, що буде отримано від продажу за зеленим тарифом, графік виготовлення енергії впродовж дня або заданого періоду.

2. ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Програмний модуль було протестовано на персональному комп’ютері, який працює на базі процесору x64 Intel Core i7 (7th Gen) та має 16 Гб оперативної пам’яті. Розроблене програмне забезпечення працює на комп'ютерах операційної системи Windows, що підходить для більшості потужних систем на сьогоднішний день.